

AD-A138 067

A SCHEME FOR THE COMPUTERIZED COMPOSITION OF URDU
NASTALIQ(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB
OH SCHOOL OF ENGINEERING A H KIZILBASH ET AL

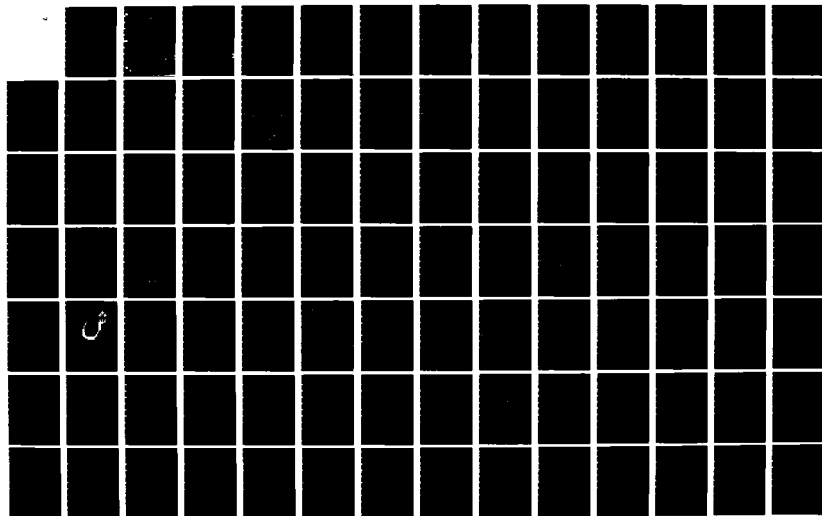
1/2

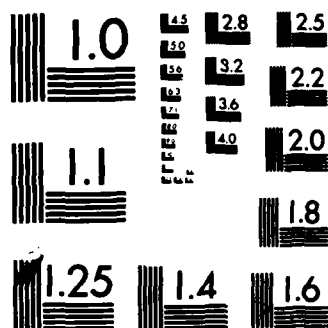
UNCLASSIFIED

06 DEC 83 AFIT/GEO/EE/83D-2

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A138067



A SCHEME FOR THE COMPUTERIZED
COMPOSITION OF URDU NASTALIQ

THESIS

Sqn. Ldr. Akeel H. Kizilbash
Capt Ronald C. Durbin

AFIT/GEO/EE/83D-2

This document has been approved
for public release and sale; its
distribution is unlimited.

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DTIC
ELECTE
FEB 22 1984

A

DTIC FILE COPY

84 02 21 179

1. Title
 2. Author
 3. Subject
 4. Date
 5. Location
 6. Distribution
 7. Availability
 8. Special

A-1

(1)

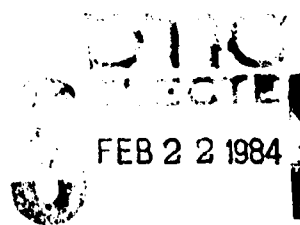


A SCHEME FOR THE COMPUTERIZED
COMPOSITION OF URDU NASTALIQ

THESIS

Sqn. Ldr. Akeel H. Kizilbash
Capt Ronald C. Durbin

AFIT/GEO/EE/83D-2



A

Approved for public release; distribution unlimited.

AFIT/GEO/EE/83D-2

A SCHEME FOR THE COMPUTERIZED COMPOSITION OF URDU NASTALIQ

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Akeel H. Kizilbash, B.E.

Ronald C. Durbin, B.S.

Sqn. Ldr. Pakistan A.F.

Capt

USAF

Graduate Electro Optics

December 1983

Approved for public release; distribution unlimited.

Preface

The purpose of this project was to make use of the computer to bring the ancient and beautiful Urdu Nastaliq script into the modern world. The work was always challenging, often frustrating, but ultimately rewarding as the approach chosen for our design proved successful. We have shown that Urdu Nastaliq, and its Arabic and Persian cousins, can be printed quickly and efficiently using reasonable computer resources. While time constraints prevented us from achieving a full implementation of the language, what we did accomplish is compelling evidence that a total solution is at hand.

We are grateful to all those who helped us in this endeavor. Our wives were a constant source of emotional support and inspiration, and Roseann did us great service as our technical editor. Mr. Ishrat H. Kizilbash got us off to a good start by sending us much critical information on the language. Mr. Chris Durbin contributed a month to develop some critical software for the expanded version we are currently implementing. Mr. Joe Hamlin's suggestions on using the VAX computer produced some startling results for which we are in his debt. Finally, our thesis advisors merit our thanks: Dr. Matthew Kabrisky, for his support, encouragement, and timely advice; Maj Walt Seward, for the numerous times he helped us choose the best of many possible

alternatives; and Lt Col Hal Carter, for his helpful critiques of the early stages of our work.

Ronald C. Durbin
Akeel H. Kizilbash

Contents

	<u>Page</u>
Preface	ii
List of Figures	vi
List of Tables	viii
Abstract	ix
 I. Introduction	 1
Background	1
Problem and Scope	6
Assumptions	6
Approach and Presentation	7
 II. Detailed Analysis.	 9
Introduction	9
Analysis of Existing Equipment	9
Selection of General Approache	11
Design Criteria	14
 III. Keyboard Development	 17
Introduction	17
Statistical Analysis of Character Occurrence	19
Key Assignment	23
Comparison to Existing Keyboards	31
 IV. Font Development	 34
Existing Urdu Fonts	34
Requirements for Developing a Standard Font	34
Pattern Recognition Considerations	39
Digitization of the Font	41
 V. Mechanization of the Font.	 47
Analysis of the Character Set	47
Standard Vertical Placement	50
EOL (End of Ligature) Characters	54
Spoilers	55
Non-Con Characters	56
Combining Ligatures	57
Inter-Ligature/Inter-Word Spacing	58
Numeral Handling	60
Text Display	60
 VI. Implementation--Program Newtext	 62
Introduction	62
Procedure Initialize	64
Procedure Process_text	71
 VII. Testing and Analysis	 83
Testing	83
Analysis	85

VIII.	Conclusions	91
IX.	Recommendations	93
	Bibliography	95
	Appendix A: Alternative Approach to Keyboard Assignment	96
	Appendix B: Physiologically Optimized Keyboard	97
	Appendix C: Example of Ligature Formation	101
	Vitae	106
	Capt Ronald C. Durbin	106
	Sqn. Ldr. Akeel H. Kizilbash	107

List of Figures

<u>Figure</u>		<u>Page</u>
1	Expanded Urdu Character Set	22
2	Frequency of Occurrence of Urdu Characters in Random Text	24
3	Areas of Finger Responsibility on a Conventional Keyboard	26
4	Preliminary Key Assignment for 30 Most Frequent Characters	27
5	Preliminary Shift Key Assignment	29
6	Final Adjustment to Keyboard Assignment	30
7	Salient Features of the Durb-Kizil Keyboard	33
8	Hand-Written Urdu Nastaliq Script	35
9	Machine Printed Urdu Scripts	36
10	Representative Shapes of Whole Characters	38
11	25 X 25 Matrix Digitization of Sheen	44
12	Comparison of Digitized Font to Hand-scribed Original	46
13	Families of Urdu Characters	48
14	Classification of Urdu Characters by Bodygroup and Diagroup	51
15	Examples of Commonly Joined Words	59
16	Program Newtext	63
17	Procedure Initialize	65
18	Display Screen Format	68
19	Procedure Process_text	72
20	Procedure Make_word	75
21	Procedure Make_ligature	77
22	Procedure Print_ligature	80

23	Procedure Print_char	82
24	Urduscribe Standard Whole Character Set	87
25	An Example of Complex Ligature Formation	89
26	Sample Sentences Printed by Urduscribe	90
27	The Physiologically Optimized Keyboard	98
28	Elements of the Whole Character Form of Chey	102
29	Elements of a Shosha Form of Zoye	104
30	Zoye-Chey Ligature	105

List of Tables

<u>Table</u>	<u>Page</u>
I. Urdu Whole Character Coding Based on Bodygroup and Diagroup	52

TR

PP 00 A138087 (U) FIELD/GROUP 000000

UNCLASSIFIED TITLE

ABSTRACT

A SCHEME FOR THE COMPUTERIZED COMPOSITION OF URDU NASTALIQ.

UNCLASSIFIED

PAGE 53

FEB 24, 1984

(U) THE POSSIBLE APPLICATIONS OF COMPUTER GRAPHICS FOR CONTROL SYSTEM DESIGN ARE CONSIDERED IN THIS THESIS. THE FUNCTIONAL REQUIREMENTS OF SUCH A SYSTEM ARE IDENTIFIED AND DISCUSSED. OTHER TOPICS SUCH AS GRAPHICS SOFTWARE, PROGRAMMING LANGUAGES, USER INTERFACE, AND SOFTWARE DESIGN ARE VITAL TO SUCH AN EFFORT AND ARE EXAMINED IN DETAIL. BASED ON THE DESIRABLE FEATURES OF DEVICE INDEPENDENT GRAPHICS, AN IMPLEMENTATION OF THE ACM/SIGGRAPH CORE STANDARD PROPOSAL IS SELECTED. TO DEMONSTRATE THE VARIOUS APPLICATIONS FOR SYSTEM ANALYSIS, A GROUP OF FUNCTIONS FOR ANALYSIS OF CONTINUOUS SYSTEMS IS IMPLEMENTED AND TESTED. SAMPLE RESULTS ARE PROVIDED AND RECOMMENDATIONS FOR FURTHER WORK ARE DISCUSSED. (AUTHOR)

POSTING TERMS ASSIGNED

APPLICATIONS OF COMPUTER
USE COMPUTERS

FUNCTIONAL REQUIREMENTS
USE FUNCTIONS
REQUIREMENTS

GROUP OF FUNCTIONS
USE FUNCTIONS

SOFTWARE DESIGN
USE COMPUTER PROGRAMS

THESES
USE THESES

CONTROL SYSTEM DESIGN
USE CONTROL SYSTEMS

GRAPHICS SOFTWARE
USE COMPUTER PROGRAMS
GRAPHICS

PROGRAMMING LANGUAGES
USE PROGRAMMING LANGUAGES

SYSTEM ANALYSIS
USE SYSTEMS ANALYSIS

USER INTERFACE
USE INTERFACES
USER NEEDS

PHRASES NOT FOUND DURING LEXICAL DICTIONARY MATCH PROCESS

UNCLASSIFIED

Abstract

A scheme for the computerized composition of Urdu Nastaliq is described. A standardized character set was defined for the Urdu language. The script was decomposed into its constituent elements which were then defined and digitized in accordance with pattern recognition principles. An efficient mapping of Urdu characters to the standard QWERTY keyboard was achieved by matching character frequency to finger agility and by applying additional pattern recognition concepts. An analysis of finger work loads was then used to make final adjustments to the layout. A set of rules for combining characters was developed, by which a sizable portion of the Nastaliq script could be composed. Rules for certain exceptional cases were not implemented. A program for displaying well-formed Nastaliq using a Vax 11/780 and a VT-550 graphics terminal was devised; the scheme is described in hardware-independent terms. Samples of text thus obtained illustrate the true composition, style, and elegance of the Nastaliq script. Recommendations for expanding the existing software to handle the entire language are included.

A SCHEME FOR THE COMPUTERIZED COMPOSITION OF URDU NASTALIQ

I Introduction

Background

Writing Systems. Writing is a system of human communication by means of conventional visible marks. Most human cultures have some form of writing or scribing. Even today the written word is the major medium for distributing information, television and radio notwithstanding.

Mechanization of Writing. Some written languages can be more easily generated, transmitted, and printed by machines (mechanized) than others. The regularity of a written language's sub-word elements determines how readily the language can be mechanized. Systems of writing derived from the Greek alphabet are perfectly regular in that the letters which make up their words always have the same shape. By contrast, most characters common to the Middle-Eastern scripts have more than one shape. The exact shape of a character is a function of its neighbors in the same ligature (sub-word element of connected characters).

Writing machines (typewriters, video terminals, etc.) were designed and have been optimized for written languages that have separated characters of regular height and width.

The major impediments to the expansion of writing machines to other systems of writing have been the diversity of character shapes, the many ways these characters can be joined to form ligatures, and the complex rules governing the positioning and spacing of characters, ligatures, and words.

Brief History of the Urdu Language. Urdu is the national language of Pakistan, but it is spoken and understood in much of the Indian subcontinent. Since many of its roots are in Arabic and Persian, the Urdu font shares many characters with these languages.

Various styles of Urdu script have evolved in the last ten centuries. The Naskh font, developed by Ibn-i-Maqlah in 932 A.D., has been adapted to print several Middle-Eastern languages including Urdu. The Naskh script is readily adapted for printing by conventional mechanical typewriters, lending the script considerable popularity in the Arab world. However, Naskh has never been popular within Urdu societies; rather, speakers of Urdu have preferred a calligraphic style of script called Nastaliq. Nastaliq was invented in the fourteenth century by Mir Ali Tabrezi when he combined the Naskh and Taliq scripts. The art of Nastaliq calligraphy flourished under the Moghul dynasty and thus by the 18th century the Nastaliq script was recognized as the standard for writing Urdu [1].

With the onset of the lithographic printing process, many efforts were attempted to preserve the character of Nastaliq. Until 1982, however, the only way to write Nastaliq was by hand. In 1981 Jamil and Saiyid devised a scheme to mechanize the Nastaliq script. With the cooperation of the Monotype Company of England, they calligraphed, digitized, and stored in a computer memory a dictionary of the seventeen thousand most commonly used fully-formed Urdu ligatures. These stored ligatures were used to form words and subsequently to be printed with a laser printer. While a significant breakthrough for printing Urdu Nastaliq, the Jamil-Monotype machine is limited to printing only those words whose ligatures are already stored in the computer's memory. Also, the process is inordinately expensive and suitable only for huge printing jobs such as newspapers and mass-circulation magazines [2].

Peculiarities of Urdu Nastaliq. Urdu characters are mostly cursive, and the ductus (path traced by the pen) is invisible only when diacritic marks are added to the basic shapes. The language is written in a script form; each word consists of one or more ligatures, individual whole characters, or a combination of ligatures and whole characters. These ligatures are made up of special representative parts of the whole character shape, called 'shoshas'. Shoshas are joined with each other according to complex but regular syntactical rules. Each character may have several different

shoshas, some of which do not resemble the parent character's shape.

The particular shosha employed is determined by the character's immediate neighbors and, occasionally, by more distant neighbors in the same ligature. The joints between shoshas must be smooth so that the whole ligature body is continuous and appears to have been made by a single ductus of the calligraphic pen. This merging process is further complicated by the varying thickness of the shoshas and whole characters along their body length.

Shoshas may appear at several different vertical levels, again depending on other neighboring shoshas in the ligature. Any given whole character will always appear on the same baseline; however, this baseline varies among different characters. The horizontal spacing between characters, ligatures, and words is variable and is a function of a separate set of rules. When shoshas are joined to form a ligature, their diacritic marks (used to distinguish a character from another having the same shape) frequently require repositioning to prevent collisions with other diacritics or with the ligature body. Horizontal overlap of separate ligatures and words is common, saves space, and is still unambiguous to the reader.

Vowel sounds in Urdu are represented by a second set of diacritics called Araab. These diacritics, if required, occur either above or below their associated shosha depending on the type of Araab.

The Urdu language is written from right to left across the page. The exception is numerals, which are written from left to right, even when they occur embedded in script.

Example of an Urdu Word. The Urdu word 'chamkili' (shining thing) is an example of a word made up of a single ligature. The ligature, in turn, consists of six characters, represented here in their whole character forms:

چ ا م ک ی ل ی

However, in the ligature these characters take on their shosha forms, as shown here:

چ م ک ی ل ی

Finally, the shoshas are joined to make the word in its final script form:

چمکیلی

Problem and Scope

The purpose of this project was to develop a technique for composing and printing high quality Urdu Nastaliq script.

Development of the technique required completion of four intermediate steps: defining an optimized Urdu character font through the use of the principles of pattern recognition; defining a keyboard for efficient text entry; formalizing the rules of ligature generation and positioning to form proper Urdu Nastaliq script; and coding the algorithms into executable programs.

While the result of this project is not a word processor, the work performed in this effort makes possible the development of a word processor for Urdu.

Software for this effort was developed on a VAX 11/780 computer, VT 550 terminal, and Toshiba P-1350 printer.

Assumptions

The major assumption upon which this thesis is based is that the tasks could be accomplished without developing any special hardware; only off-the-shelf equipment would be used.

Approach and Presentation

The approach followed in this thesis was to develop, and to verify by demonstration, a set of syntax-based computer programs which, when applied to strings of input characters in the Urdu language, generates well-formed ligatures. This method was chosen over an alternate approach which uses a dictionary of pre-defined ligatures.

Development of the rule-driven compositor described here followed a detailed analysis of the problem and required the accomplishment of four major tasks. The detailed analysis is described in Chapter II, while the major tasks were performed sequentially and are described in Chapters III through VI.

Chapter II is the detailed problem analysis, and consists of three parts. First, existing equipment which perform tasks similar to the compositor are analyzed. Next, the two possible approaches to the problem are discussed in general terms and the selection of the approach used in this thesis is justified. Finally, a set of design criteria for the compositor is defined.

The first step was development of a text entry technique; this process is detailed in Chapter III. Keyboard layout, including Urdu to ASCII character conversion, is described.

Chapter IV describes the procedures followed in designing a standard character font for the Urdu language. The process for defining every whole character, shosha, and diacritic in terms of a dot matrix is described. This definition allows characters to be displayed on a video terminal or printed on a dot matrix printer.

The third and most substantial task is that described in Chapter V. In this chapter, rule-based ligature generation is detailed.

Chapter VI details the coding of the text processing, ligature generating, text display, and text printing algorithms in a program called Newtext.

Chapter VII describes the criteria and process used for testing the software described in Chapter VI, as well as analyzing the results of the previous chapters.

Conclusions suggested by the analysis of the compositor approach and implementation are summarized in Chapter VIII.

Chapter IX contains recommendations which, if implemented, should permit the compositor to satisfy fully its functional specification and design criteria.

II Detailed Analysis

Introduction

This chapter describes past and present attempts to mechanize the Urdu language. Following this introduction, machines which are currently available for printing Urdu are described. The description includes details on both a Naskh typewriter and a computer-driven Nastaliq phototypesetter. The next section defines, then compares, two general approaches to printing Urdu Nastaliq with machines. The advantages and disadvantages of the two techniques, termed the ligature-based and rule-based approaches, are discussed and form the basis for selecting the rule-based method of this thesis. Having selected an approach to solving the problem, the final section lists design criteria for the Urdu text compositor. The design criteria consist of a functional specification and a set of acceptance criteria.

Analysis of Existing Equipment

Urdu Naskh Typewriter. Naskh is a script used widely in the Middle East for writing Arabic and Urdu. A mechanical typewriter for Arabic Naskh has been available for many years and has become common in most of the Arab world. A version of the Arab Naskh mechanical typewriter was slightly modified for Urdu Naskh. This typewriter is similar to the Arab original, even retaining some peculiarities of the Arab

script not used in Urdu. In contrast to other countries, however, the government of Pakistan has discouraged the use of the Naskh script in schools, supporting instead the handwritten script which is Nastaliq. Nevertheless, IBM has foreseen sufficient market potential to introduce in 1983 an electric version of the Urdu Naskh typewriter [3].

The Naskh script is simple compared to Nastaliq, but it still required some modifications to permit adaptation to a typewriter. The major simplification was the "averaging" of the shosha forms of the characters so that a single shape could be used in many situations. Due to this forced standardization, the shoshas in a ligature of Naskh join with each other at inappropriate levels. The font that has resulted is suitable for mechanization but bears little resemblance to Nastaliq.

The Urdu Naskh keyboard is physically similar to the common QWERTY keyboard. Shosha forms of characters are assigned to individual keys, while the whole form of each character is obtained by simultaneously depressing the shift key and the key corresponding to the character's shosha form. The placement of characters seems to be arbitrary; no account was taken of the relative frequencies of occurrence of each letter or of the significantly different capabilities of each finger. The result is that typing on the Urdu Naskh typewriter is slow, inefficient, and tedious.

Urdu Phototypesetter. In 1982 work was completed on the first phototypesetter capable of printing a variant of Urdu Nastaliq called Noori Nastaliq [2]. The phototypesetter accomplishes its task by making words from pre-defined ligatures contained in a dictionary of such entries. Each ligature of the tens of thousands that are stored was drawn by hand, photographed, and digitized. The result is a relatively uniform and readable font. The machine is not without its problems, however. The Jamil-Monotype phototypesetter can only print words whose ligatures are in its dictionary; consequently, many words, notably proper names, cannot be written by the machine. Further, the phototypesetter cannot print Araabs, making pronunciation of words difficult for those not already fluent in the language. Also, the absence of Araabs makes it impossible to completely express the pronunciation of new words and proper names. Finally, the machine is a phototypesetter--its use for small printing jobs is impractical and the approach upon which it is based is not transferrable to machines suitable for general use such as typewriters and small computer-based word processors.

Selection of General Approach

Careful consideration resulted in two possible approaches to the problem of composing Urdu Nastaliq. The technique used by the Jamil-Monotype phototypesetter is referred to as the ligature-based approach. Another technique

was conceived as part of this thesis and is called the rule-based approach.

Ligature-Based Approach. In this scheme, ligatures are printed from templates stored in a dictionary of complete ligatures. An operator enters a series of keystrokes, with each ligature terminated by an operator-inserted command. The characters of the ligature are translated by the computer into an address at which the description of the entire ligature is to be found.

While this approach is conceptually simple, it has several serious disadvantages. First, only ligatures already stored in the computer's memory may be printed. As a result, every ligature which could conceivably be used, however infrequently, must be stored. Second, this technique is extremely wasteful of memory, since every character of every ligature must be stored. This tremendous redundancy makes the dictionary containing the ligatures a cumbersome data structure requiring large seek times for individual entries. For example, the set of ligatures containing four or fewer characters would require storage of over eight hundred thousand unique entries.

Rule-Based Approach. The concept of the rule-driven compositor is to develop a set of rules which, when applied to strings of input characters, generate well-formed ligatures. The ligatures are made by joining the appropriate

shosha forms of the input characters according to the rules contained in the dictionary. Besides indicating which shoshas to use, the dictionary entries give positioning data for the shoshas. Using this approach, only a minimum set of character building blocks must be stored in memory, along with the rules for connecting the pieces. While the set of all possible combinations of shoshas is large, many situations can be handled by a set of rules governing two character ligatures. The number of rules required is on the order of one thousand; moreover, the rules apply by extension to ligatures of more than two characters. Exceptions which occur require definition of a specific third-order rule. These exceptions are few and have been noted; however, the rules could not be developed because of lack of time. Exceptions to the third-order ligature rules are rare and can be handled on an individual basis. Thus, even though the rule-based approach is initially more difficult, ultimately this technique is both more flexible and a complete solution. That is, any ligature can be constructed, even those without meaning or those not yet defined.

Selection of Approach. The rule-based dictionary approach was chosen for this thesis. The inherent limitations of the ligature-based approach far outweigh its apparent simplicity, while the power and flexibility of the rule-based approach make the conceptual difficulties worth overcoming.

Design Criteria

General. The result of this effort shall be one or more computer programs and associated databases (software). This software, when executed on suitable hardware, shall perform the functions of an Urdu type compositor. The specific attributes of such a compositor are described in the functional specification. The software and hardware shall satisfy the specific requirements contained in the acceptance criteria.

Functional Specification. An Urdu type compositor is a machine which composes Urdu Nastaliq text. Such a machine performs four distinct operations. First, text entered by an operator is accepted from a text entry device. Second, the machine composes the input text as characters, ligatures, and words in accordance with the rules of written Urdu. Third, the compositor displays the formatted text to provide visual confirmation of input and output. Last, the machine provides a means of creating a paper copy of the output text.

Acceptance Criteria. The criteria listed here form the basis for accepting the compositor. These criteria shall be translated into tests, and the compositor shall be judged to satisfy the functional specification only when all tests are satisfactorily completed.

1. Text Entry

a. The text entry device shall permit the introduction of all necessary Urdu elements, including the set of whole characters as defined, numerals, Araabs, and punctuation.

b. The assignment of Urdu elements to key positions shall be based on the criterion of user efficiency.

c. Unused (unassigned) keys shall be ignored by the software.

d. A file of user-input text shall be created.

2. Text Composition

a. Words shall be recognized by the occurrence of operator-inserted spaces.

b. Single characters followed by spaces retain their whole character forms.

c. Ligature formation, by the selecting and joining of specific shoshas, shall take place automatically.

d. Ligature formation shall be rule-driven.

e. Horizontal spacing between characters, ligatures, and words shall be determined automatically.

f. Vertical positioning of characters and ligatures shall be determined automatically.

3. Text Display

a. Whole characters originating at the text entry device shall be displayed.

b. Lines of text shall be displayed from top to bottom of the display screen.

c. Text shall be displayed right to left.

d. Numerals shall be displayed left to right.

e. Word-wrap shall be automatic.

f. There shall be a one-to-one correspondence between characters on the display and characters on the printed output.

4. Text Printing

a. Printed text shall be legible and compact.

b. Characters and shoshas shall retain their proper shapes and proportions when printed.

c. Text shall be right-justified.

III Keyboard Development

Introduction

Topic and Presentation. The keyboard, as text entry device, deserves special consideration for two reasons. First, the keyboard is the user's primary interface with any word processing system; and second, the Durb-Kizil keyboard described here, in contrast to existing standard keyboards, has been designed in accordance with the principles of pattern recognition and human physiology. This introduction will briefly describe the evolution of the "standard" English keyboard, the QWERTY. Succeeding sections will trace the development of the Durb-Kizil Urdu Keyboard.

Early Keyboards. Early keyboards bear little resemblance to each other. The first keyboards differed in such fundamental areas as the total number and relative placement of the keys. As mechanical designs converged, these differences disappeared and all typewriters placed their keys in similar horizontal rows; the assignment of letters to key positions, however, was still unique to each machine [4].

The present English-language standard keyboard is referred to as the QWERTY. The name is derived from the left-to-right sequence of the first six letters of the top row of keys. The QWERTY keyboard uses three rows of keys for

the alphabetic character set. To overcome mechanical constraints derived from the keying mechanisms, the three rows of keys were arranged in a left-leaning staircase fashion. This placement forces the typist's fingers to move diagonally, instead of in a natural up and down motion. Incredibly, this design has been passed on to present-day electric typewriters where the original mechanical constraints do not exist.

The unfortunate letter-to-key assignment of the QWERTY is a legacy of Sholes and his associates who developed the arrangement empirically to eliminate type-bar clash in the crude mechanical typewriters of the day. Sholes began with a basically alphabetical ordering (still preserved in the middle row's DFGHJKL sequence), then switched pairs of letters until type-bar jamming became sufficiently infrequent [5].

The Sholes machine became a commercial success and was widely copied, making the QWERTY layout the de facto standard. Since that time the vast investment in equipment and training on the QWERTY keyboard has proved to be an effective barrier to the adoption of the many improvements developed since Sholes, even though compelling reasons argue for such improvements.

The problems of the QWERTY layout are its lack of consideration of human physiology and its inefficient letter-to-key assignment. The incorporation of straight rows of

keys, for example, forces the wrists to be bent constantly and requires the typist's arms to be held closely by the sides, resulting in unnecessary fatigue. The keyboard takes no notice in its letter placement of the frequency with which letters occur nor of the dexterity of the finger assigned to each letter. This oversight has led to the use of the same finger for both "d" and "e", even though those letters occur together with great frequency, and the use of the left small finger (the weakest finger) for the common letter "a". One result: in a right-handed world, the left hand works harder than the right hand. Now that electric and electronic keyboards have obviated the original mechanical problems, the poor key arrangement and letter assignments are proving to be the limiting factors in typing comfort, speed, and accuracy.

Statistical Analysis of Character Occurrence

Selection of Text Data. The design of an optimized keyboard requires knowledge of the frequency of character occurrence. The occurrence of words, and thus characters, is a function of the particular piece of text sampled. However, if a large number of small pieces of text on random topics is considered, then the overall effect due to the repetition of keywords is minimized. Therefore, in order to obtain unbiased character occurrence data, text pieces from various Urdu books, magazines, and newspapers were collected for statistical analysis.

Determination of Independent Characters. Before any collection of data could be attempted, a decision had to be arrived at on the total number of independent character keys required for written Urdu. For maximum efficiency, the number of keys used by the operator should be kept to the barest minimum. On an English keyboard such a decision is trivial, as there are a fixed number of characters in the alphabet and their shapes only change between upper and lower case.

In the case of Urdu, although there is no concept of upper or lower cases, each character can take on a variety of shapes, called shoshas, depending upon its usage in a ligature. A character is always represented by a unique shosha in a particular ligature except for certain cases where more than one shosha can be validly used in the same ligature; in this case the two shoshas actually make two different words. These exceptions were handled by adding the alternate character forms as if they were independent characters in their own right. This selection of unique shoshas for ligatures allows the generation of a definite set of rules of ligature formation. The result is tremendous operator relief, as only whole characters need be typed in. The typist need not decide which shosha is required as the computer handles all these decisions.

Independent Forms of Characters. An in-depth search of the Urdu script revealed that exceptional forms appeared only for three characters. The three special characters include modified versions of Alif, Noon, and Ha. This required the regular set of 37 whole characters to be increased to 40. The new character set is illustrated in Figure 1.

Alif can appear in a special form called 'Alif Mad Zabbar' (AMZ), the difference in shape being the addition of a tilda over the top of the original Alif form. The next exception is that of Noon, which can appear in the form of 'Noon Ghunna' (NG). The difference in appearance of the NG is the absence of the single 'nukta' (a type of diacritic mark) of the regular Noon. The last exception is that of Ha. It is especially exceptional because it can appear in three different forms. The more often appearing variant is that of the 'Do Chashmi Hay' (DCH), meaning Hay with two eyes (as the shape appears to be). The other variant of Ha is unnamed and occurs only in the word Allah, which is an Arabic word commonly used in Urdu. The word Allah and this form of Ha obey the rules of Arabic script not common to Urdu. Therefore, the word Allah was dealt with separately.

At this stage an alternative approach to key assignment was considered which could have reduced the total number of keys by more than twenty-five percent. However, this approach was not adopted because its advantage of a relatively small

No.	Whole Char. Shape	Name	Nearest Sound	No.	Whole Char. Shape	Name	Nearest Sound
1.	ا	Alif	A	21.	ض	Zoad	Z
2.	ب	Bay	B	22.	ط	Toye	T
3.	پ	Pay	P	23.	ظ	Zoye	Z
4.	ت	Tay	Soft T	24.	ع	An	A
5.	ٹ	Ttay	T	25.	غ	Ghen	Gh
6.	ث	Say	S	26.	ف	Fay	F
7.	ج	Jeem	J	27.	ق	Kkaf	Q
8.	چ	Chey	Ch	28.	ک	Kaf	K
9.	ح	Hay	H	29.	گ	Gaf	G
10.	خ	Khay	Kh	30.	ل	Laam	L
11.	د	Dal	Soft D	31.	م	Meem	M
12.	ڈ	Ddal	D	32.	ن	Noon	N
13.	ذ	Zal	Z	33.	و	Vow	V
14.	ر	Ray	R	34.	ہ	Ha	H
15.	ڑ	Rray	Hard R	35.	ع	Humza	Aae
16.	ز	Zay	Z	36.	ی	Choti ye	E,Y
17.	ڑ	Yay	Y	37.	اے	Bari Ye	Ay
18.	س	Seen	S	38.	آ	AMZ	Aa
19.	ش	Sheen	Sh	39.	ھ	DCH	H
20.	ص	Soad	S	40.	ں	NG	Soft N

Figure 1. Expanded Urdu Character Set

keyboard was outweighed by the ultimate increase of typing effort. The details of this approach are given in Appendix A.

Collection and Analysis of Data. A sample of more than six thousand words (over twenty-three thousand characters) was assimilated from various pieces of Urdu text. A computer program was written to process the raw data for the following parameters:

(a) Cumulative frequency of occurrence of each character for every two thousand characters of data.

(b) Succedance of every character with all characters including the space between words.

The total frequency of occurrence of each character is illustrated in a histogram in Figure 2. The statistics derived from the data sample were judged sufficiently representative of the language to serve as the basis for making character-to-key assignments.

Key Assignment

The key assignments were completed in three phases. The first phase was based on purely statistical results, while the second phase utilized the principles of pattern recognition. In the third phase the assignments were finalized after considering overall finger workloads.

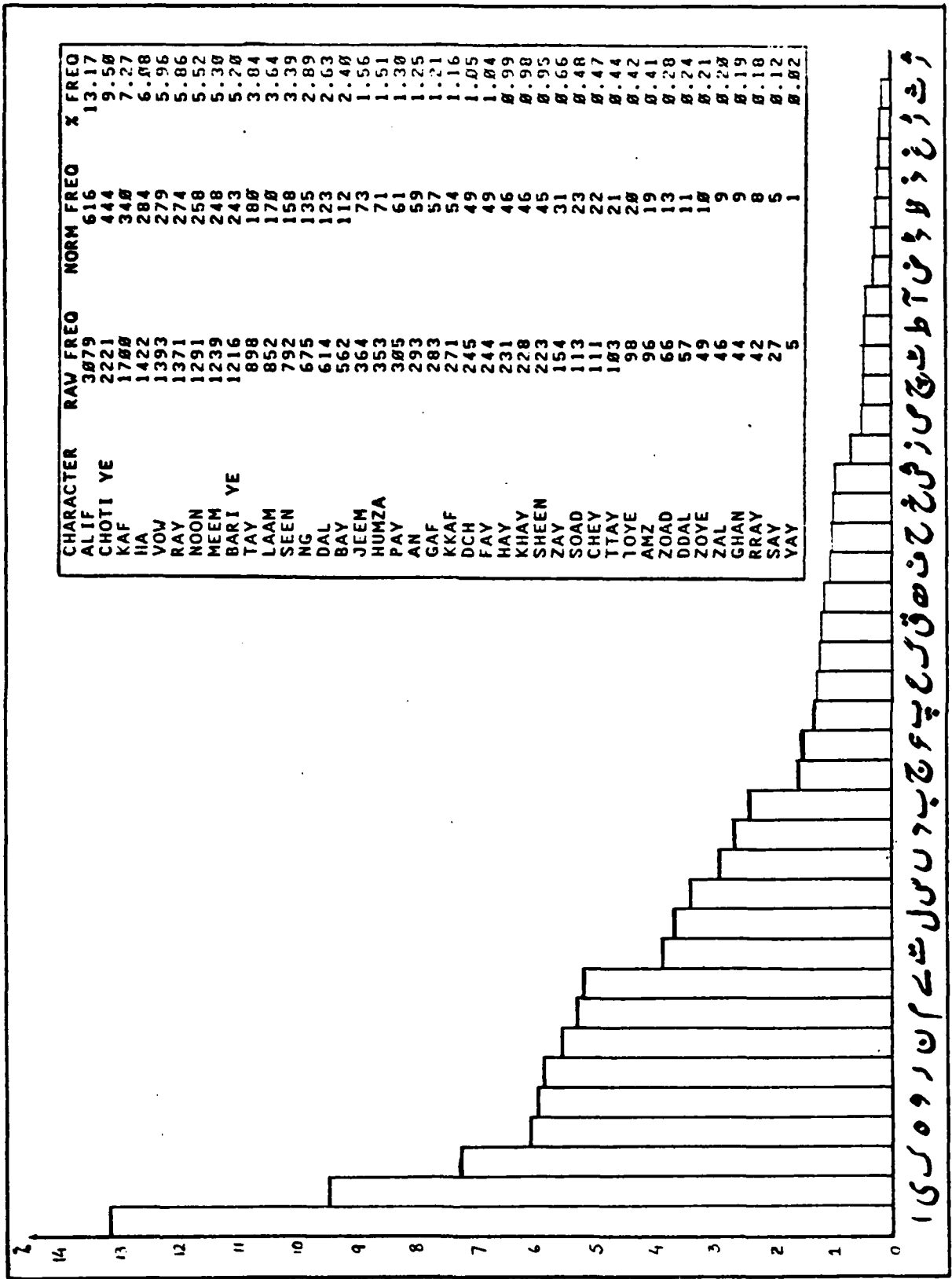


Figure 2. Frequency of Occurrence of Urdu Characters in Random Text

Finger Agility. On a conventional keyboard there are three rows of character keys. The rest positions for the fingers is on the keys of the middle row, whereas the thumbs stay on the space bar. The division of finger responsibility on the conventional keyboard is shown in Figure 3.

A finger's gymnastic potentials depend upon its agility and strength. Earlier studies [4] have suggested the following ordering of finger capabilities on a keyboard:

- (i) index finger
- (ii) middle finger
- (iii) ring finger
- (iv) small finger

As the majority of people are right-handed the right fingers were favored over their counterparts on the left hand.

Frequency-Agility Match. On a conventional keyboard the keys of the middle row, on which the fingers rest, are the best candidates for the most frequently occurring characters. Knowing the frequency of occurrence of the character set and the order of the finger capabilities, the following assignment was systematically made for the most frequent eight characters. These assignments are shown in Figure 4. Although the 'g' and 'h' keys are on the middle row and are operated by the index fingers, they require deterministic finger re-positioning and thus lose their favored choice. As the lateral movement of the index finger to

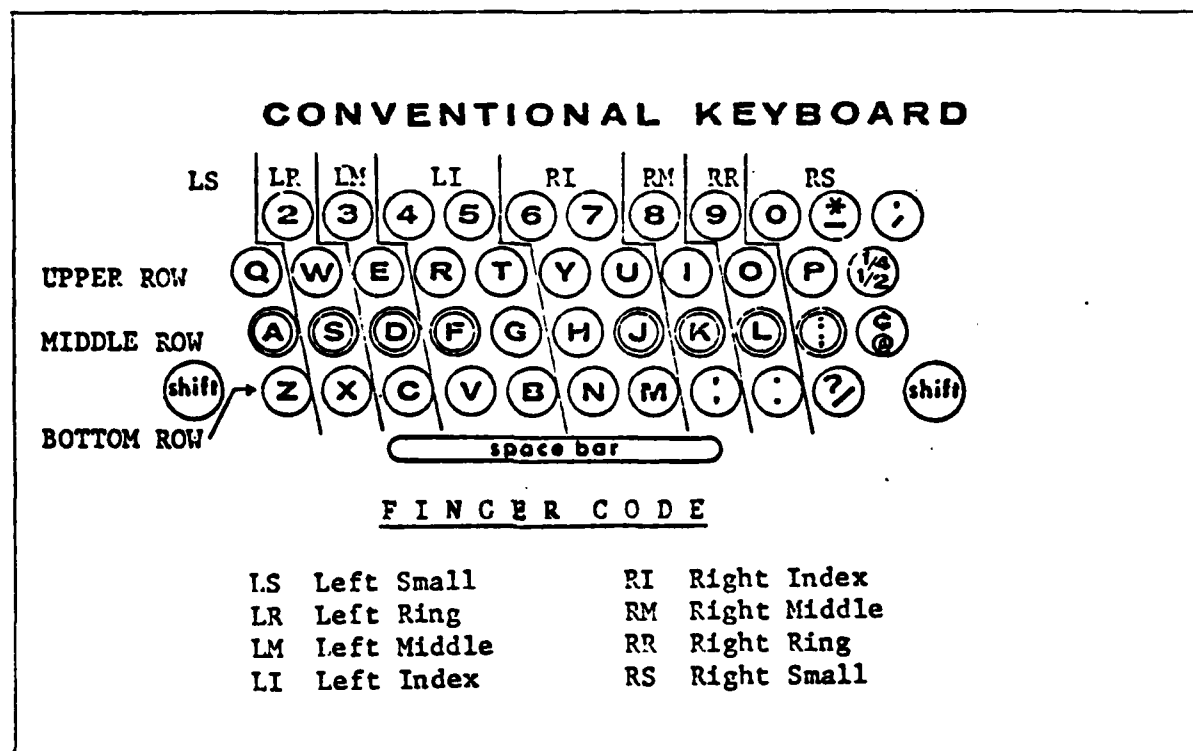


Figure 3. Areas of Finger Responsibility on a Conventional Keyboard

operate them is less than the movement to reach a different row, they are still favored over keys on other rows. Thus the next two assignments were carried out for the 'g' and 'h' keys. The choice of characters for these keys can also be seen in Figure 4.

A micro-motion study revealed that the character keys of the upper row are usually reached by stretching the fingers from their rest positions on the middle row; to reach the bottom row, however, the whole arm has to be pivoted back. This motion of the arms is time consuming and fatiguing. Thus the upper row was preferred over the bottom row

Urdu Char.	Freq.of Occurrence	Key Assign.	Urdu Char.	Freq.of Occurrence	Key Assign
Alif	13.17%	j	Jeem	1.56%	w
Choti ye	9.50%	f	Humza	1.51%	p
Kaf	7.27%	k	Pay	1.30%	q
Ha	6.08%	d	An	1.25%	y
Vow	5.96%	l	Gaf	1.21%	t
Ray	5.86%	s	Kkaf	1.16%	m
Noon	5.52%	;	DCH	1.05%	v
Meem	5.30%	a	Fay	1.04%	,
Bari ye	5.20%	h	Hay	0.99%	c
Tay	3.84%	g	Khay	0.98%	x
Laam	3.64%	u	Sheen	0.95%	.
Seen	3.39%	r	Zay	0.66%	/
NG	2.89%	i	Soad	0.48%	n
Dal	2.63%	e	Chey	0.47%	z
Bay	2.40%	o	Ttay	0.44%	b

Figure 4. Preliminary Key Assignments for 30 Most Frequent Characters

in the key assignment process. The same logic developed for key assignments on the middle row was implemented on these rows as well. The assignment of characters to the upper and bottom rows is shown in Figure 4. In this manner the thirty most frequent characters were assigned to the thirty available keys. The remaining ten characters then had to be co-assigned to the existing keys using the shift function. The

operation of the shift key is an additional task for the operator which causes distraction in the rhythmic and smooth flow of the typing operation; however, no easy alternative exists on the conventional keyboard. By assigning the least occurring characters to the shift mode, the shift key is used for less than two and one-half percent of typed characters.

Application of Pattern Recognition. In order to co-assign two characters to a key, pattern recognition factors were taken into account. By assigning similar characters to the same key it becomes easier to represent, search for, and commit to memory the position of both characters. The concept of families of characters described in Chapter V and illustrated in Figure 12 provides a basis for co-assignments.

The ten remaining characters requiring assignment are shown in Figure 5. Of these ten characters AMZ, Zoad, Ghan, Rray, and Yay were co-assigned to the most commonly occurring member of their respective families. Rray was favored over Yay in this co-assignment due to the higher occurrence of Rray. Together Toye and Zoye, which are the only two members of the Toye family, needed a separate key. Ttay, the least occurring character, which was initially assigned a key by itself was co-assigned with Tay (highest frequency in the Bay family). The key thus freed was assigned to Toye and Zoye. Zoye was assigned to the shift mode as it occurs less than Toye. Ddal and Zal of the Dal family have only Dal assigned

Urdu Char.	Freq.of Occurrence	Key Assign.	Urdu Char.	Freq.of Occurrence	Key Assign
Toye	0.42%	b	Ghan	0.19%	Y
AMZ	0.41%	J	Rray	0.18%	S
Zoad	0.28%	N	Say	0.12%	O
Ddal	0.24%	z	Yay	0.02%	?
Zoye	0.21%	B	Chey	0.47%	W
Zal	0.20%	E	Ttay	0.44%	G

Figure 5. Preliminary Shift Key Assignments

an independent key and therefore only one character of the Dal family can be co-assigned. To uphold the pattern recognition principle it was preferred to vacate another key for the third member of the Dal family rather than to co-assign it to a member of another family. For this purpose Chey was selected, as Soad already shares its key with Zoad. Chey was co-assigned with Jeem, Ddal was placed on the key previously used by Chey, and Zal was co-assigned with Dal as it occurs less than Ddal. With the case for Ttay settled, Say was co-assigned with Pay. As Ttay and Chey occur with frequencies of only 0.44% and 0.47%, their repositioning did not degrade the keyboard's efficiency to any noticeable extent. This completed the second phase of the keyboard layout.

Final Key Assignment. At this stage the overall finger workload was analyzed. The result was as expected and desired: the right hand did more work than the left, and the

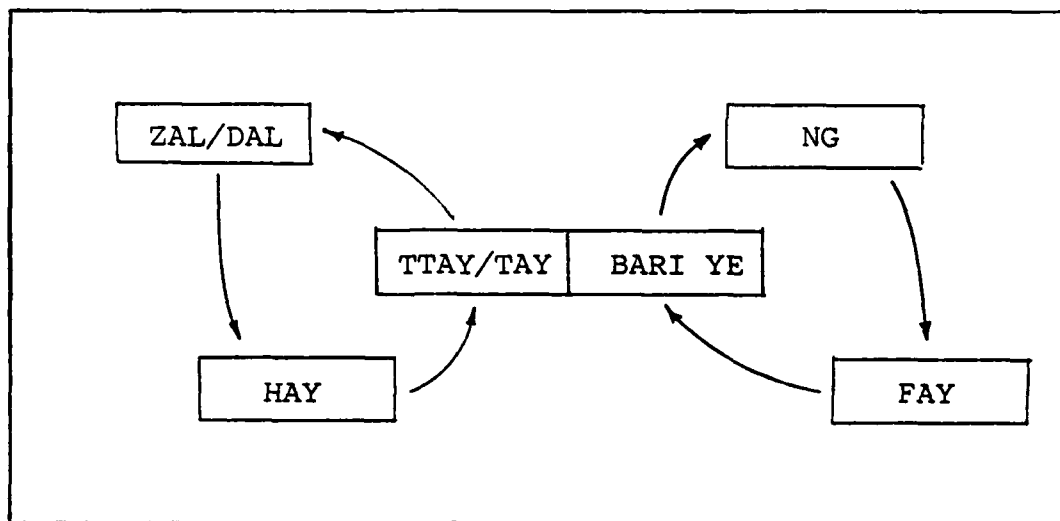




Figure 6. Final Adjustment to Key Assignment

fingers were tasked in accordance with their capabilities. However, due to vast differences of character frequency, the index fingers of both hands were heavily overworked (26.31% for the right index and 20.15% for the left index finger). Although these are the most capable fingers, their overloading meant not only tiring them easily but also making them move frequently to other rows. Thus a reassignment was needed in which the index fingers were relieved of some of their load by moving it to the middle fingers. The final changes to the keyboard layout are shown in Figure 6. The bases for these changes were transfer of index finger loading, as mentioned above, and increased favor of the top row over the bottom row.

Comparison With Existing Keyboards

Both the existing machines which print Urdu (i.e. the mechanical Naskh typewriter and the Monotype ligature-based system) have different keyboards. Some of the dissimilarities result from the different number of independent character keys each system uses, owing to the different methods of composing the script. But even the assignment of common characters appear to have been done at random, disregarding the criteria discussed earlier. Moreover the large number of key operations required increases the operator's task and reduces efficiency.

The Naskh Keyboard. In this mechanical system most of the keys have two shapes for each character. An averaged-out shape of the shoshas appear in the regular mode, while the whole character shape takes the shift mode since it occurs less frequently. Nevertheless, the whole shape does appear very frequently; the result is that the shift key is used very often, thereby severely reducing the efficiency. This frequent use warrants a favourable position for the shift key; but perhaps due to mechanical constraints the shift key is still handled by the smallest fingers of either hand. It is surprising to note that due to the Arabic legacy of Urdu characters like  and , which are commonly used in Arabic but have no usage in Urdu, are still able to find a place on this keyboard.

The Monotype Keyboard. In the case of the Monotype keyboard there are more than forty character keys and a large number of other functional keys the purpose of which are not known because the information was not available. The character keys of the same family appear more closely grouped as compared to the mechanical typewriter.

Comparison. A one-to-one comparison of the Durb-Kizil keyboard with the mechanical and Monotype keyboards is unfruitful since the total number of keys is different for all the keyboards. But it is evident that due to the least number of keys on the Durb-Kizil keyboard and because the key assignments are based on scientific criteria, this layout should far exceed the other keyboards in efficiency. Moreover, because it requires no repetitive use of function keys like the positioning keys or the end of ligature keys, it will ensure the smooth flow of the typing process. The salient features of the Durb-Kizil keyboard are summarized in Figure 7.

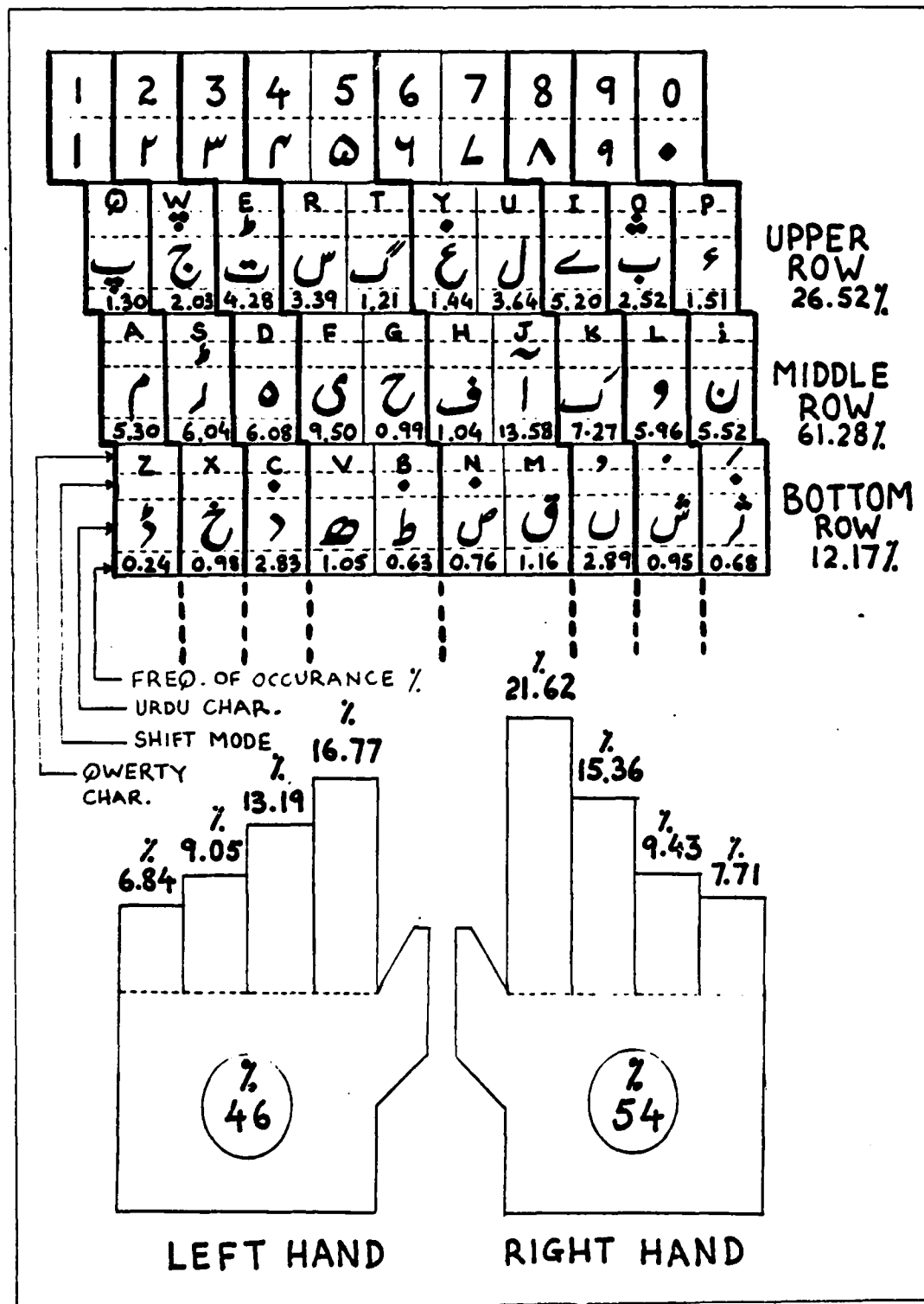


Figure 7. Salient Features of the Durb-Kizil Keyboard

IV Font Development

Existing Urdu Fonts

As of now the most common way to write Urdu is by hand. The mechanical typewriter prints the Naskh style, which is not popular, and the Monotype system prints Noori Nastaliq but is only used by one newspaper. Although these machines to type Urdu with a consistent font do exist, nearly all Urdu Nastaliq is written by hand, the font differing from scribe to scribe. The scribe, while hand writing the font, enjoys the freedom of squeezing-in and horizontally overlapping characters, ligatures, and even words when space limitations occur. He also exercises discretion in moving the positions of diacritic marks to the empty spaces in a complex ligature. Of the various fonts written by scribes, the most commonly used is shown compared in Figure 8 to two other versions of the hand-written font. The Naskh font of the mechanical typewriter and the Noori Nastaliq of the Monotype system are shown in Figure 9.

Requirements for Developing a Standard Font

Hand calligraphed character sets by different scribes have variations in the shapes, relative thicknesses along body lengths, and comparative sizes of characters. In order to mechanize the language, the first task was to develop

اسی طرح دقت میں بھی کئی فیصد کی بچت ہوگی۔ مگر ایک کتاب کی کتبت میں مہینوں بلکہ برسوں لگ جاتے تھے۔ کیونکہ ایک ہی کاتب اور ایک ہی خط چاہیے۔ ظاہر ہے کہ انسان اور مشین میں تو مقابلہ وقت کم ہی لگتا ہے۔ پھر جب انسان کی انگلیوں اور کمپیوٹر کی تیز رفتار شعاع میں مقابلہ ہو، تو انسان کی انگلیاں بہت پیچھے رہ جائیں گی۔ انسانی دماغ بھی تھک جاتا ہے۔ انگلیاں بھی تھک جاتی ہیں۔ اس تحریر میں یکسانیت بھی نہیں ہوتی۔ انگلیاں اور آنکھیں تھکتی جاتی ہیں۔ تحریر کی رفتار اور معیار میں فرق آتا جاتا ہے۔ کمپیوٹر کے ساتھ یہ معاملہ نہیں ہے۔

A popular version of hand-scribed Urdu Nastaliq
(Adapted from Noori Nastaliq, Computer se Kitabat, Elite Publishers, Ltd., Karachi)

دس گئے

اجزاء :- دو دھ ایک سیر، چینی آدھ سیر، مارک ایسٹ ایک پٹلی کے برابر چادروں کا آٹا آدھ پاؤں
توکیمب :- دو دھ کر گدھی میں ڈال کر چوٹے پر چڑھا دیں جب پکنے لگے تو اس میں مارک ایسٹ
ایک چنگی برابر ڈالیں جب تو دھ پیٹ جائے تو تالیں اور اس کو بائیک کپڑے میں ڈال کر
باندھ کر کہیں لٹکادیں تاکہ پانی نہ چڑھائے اور بنیر پانی رہ جائے۔
اب اس بنیر میں چادلوں کا آٹا ملا لیں اور ایک صاف تسلا یا لکڑی کا صاف تختہ لے کر
اس پر ماتھ کی پتیلی سے آٹا ملا لیں کہ چادلوں کا آٹا اور بنیر مل کر بالکل یکجان ہو جائے۔ اس کو
ایک گھنٹہ تک ہوسے ہوسے اسی طرح طے ریں جس طرح ہوتوں کے لئے زیورہ ملا جاتا ہے
اب اسکی چھوٹی چھوٹی گولیاں بنالیں، اور رکھ لیں۔

چاکلیٹ انڈسٹری کیلئے بیرونی ریپر کی تیاری میں خاصی مقدار
میں اچھی کوالٹی کے چھپے ہوئے کاغذ کی کمزورنگ مشین استعمال ہوتی ہے۔
ریپر تیار کرنے والی زیادہ تر مشینیں یا تو پہلے سے کٹے ہوئے
بیرونی ریپر استعمال کرتی ہیں یا پھر ریل کی شکل میں جو کہ فوٹو ایکٹرک
سیل رجسٹریشن سے کنٹرول ہوتا ہے۔ اسی طرح منڈی کر شیت
فیڈ آؤٹ اور گریو ریل دونوں کے لئے مبنی نش موجود ہے۔
ریپر تیار کرنے کے دو طریقے رائج ہیں (۱) لفٹ کی شکل
کا ٹول بنیر جس میں اندرونی فوٹل ریپر کو فولڈ کر دیا جاتا ہے لیکن
سیل نہیں کیا جاتا بلکہ بیرونی ریپر کو گوند سے سیل کر دیا جاتا ہے
یہ دونوں ریپر ساتھ ساتھ فولڈ کئے جاتے ہیں۔ (۲) اندرونی ریپر
کو گریو بنیر سے سیل کر دیا جاتا ہے اور بیرونی ریپر کو لفٹ کی شکل
دے دی جاتی ہے۔

Two other versions of hand-scribed Nastaliq
(Left sample adapted from Khana Pakana, Hafiz Ayatulla, Taj Company, Ltd., Lahore; right sample from Printer's, July-August 1982 edition, Karachi)

Figure 8. Hand-written Urdu Nastaliq Script

مگر ماں جی یہ سن کر سکتہ میں سینہ دوسرے دن دونوں لڑکیاں تھیں اور کھانا جی کیں نا ہمارے گھر ہمارے دادا جان نے ہین بہت اچھے ہین ہاں بہت اچھے ہین ماں جی نے کھوئے سے لہجے میں کھا۔ شہلانے پوچھا آپ جاتی ہیں انہیں اونہیں بس یونہی سب کچھ رہے ہیں نابینا سی لٹے میں نے بھی کھا۔ اچھا چلیں نا انہیں ہمارے گھر چلینہیں بیٹا تم لوگ جاتو۔ نہیں یہی چلیں وہ دونوں ضد کرنے لگیں اور ماں جی جانا نہیں چاہتی نہینا اچھا ہم جارہے ہیں ابو اور دادا کو۔ ہاں لے کیں گی۔ نہیں مانجی نے خدا سے دعا مانگی اے مالک میری دعا قبول کرنا کہ میرا وعدہ پورا ہو جائے اے خدا اچھے اپنے پاس بلالے تاکہ میں اپنے عہد اپنے راز کو اپنے ساتھ ہمیشہ کے لئے دفن کر دوں اور شاید وہ قبولیت کی گھڑی نہیں کہ ماں جی نے نماز پڑھنے کی تیاری کی اور وہیں مسجد میں جان جان قریب کے سپرد کر دیا۔ ادھر ماں جی نے داعی اجل کو لبیک کہا۔ ادھر نجل حسین نے اپنے والد افضل حسین کو ماں جی کے گھر سے لے کر اسی اثناء میں باہر شور ہوا تو نہ چلا کہ ماں جی مر گئیں تمام لوگ ماں جی کے گھر ہا گیا۔ نہیں نجل حسین کے بیوی بچے اور اس کا باپ افضل حسین یہی تھا

Naskh script by a mechanical typewriter

دینیں اور جمل حسین کو ان پرست احمد تھا اگر کیں جانا ہوتا تو اپنی دونوں بیٹیوں کو ماں جی کے حوالے کر جاتے اور کہتے ہم کیں جارہے ہیں آپ بچہ جیوں کا دھیمان رکھئے گا۔ اس طرح وہ سب پر محبتوں کے غم نے نچھاور کرتی رہیں۔ ایک دن جمل حسین نے بتایا کہ ماں جی میرے والد مرے بعد سماں آ رہے ہیں۔ گھر ماں جی یہ سن کر سکتہ میں آئیں۔ دوسرے دن دونوں لڑکیاں آئیں اور کہاں جی آئیں نا ہمارے گھر ہمارے دادا جان آئے ہیں۔ بہت اچھے ہیں۔ "ہاں بہت اچھے ہیں ماں جی نے کھوئے سے کچھ میں کھا۔ شہلانے پوچھا آپ جاتی ہیں انہیں۔ او نہیں بس یونہی سب کہ رہے ہیں نابینا سی لٹے میں نے بھی کھا۔ اچھا چلیں نا انہیں ہمارے گھر چلیں۔ میں بیٹا تم لوگ جاؤ۔ نہیں آپ بھی چلیں وہ دونوں ضد کرنے لگیں اور ماں جی جانا نہیں چاہتی تھیں۔ اچھا ہم جارہے ہیں ابو اور دادا کو۔ ہاں لے کیں گی۔ نہیں میں ماں جی نے خدا سے دعا مانگی اے مالک میری دعا قبول کرنا کہ میرا وعدہ پورا ہو جائے اے خدا اچھے اپنے پاس بلالے تاکہ میں اپنے عہد اپنے راز کو اپنے ساتھ ہمیشہ کے لئے دفن کر دوں۔ اور شاید وہ قبولیت کی گھڑی تھی کہ ماں جی نے نماز پڑھنے کی تیاری کی اور وہیں مسجد میں جان جان قریب کے سپرد کر دی۔ ادھر ماں جی نے داعی اجل کو لبیک کہا۔ ادھر جمل حسین نے اپنے والد افضل حسین کو ماں جی کے گھر سے لے کر اسی اثناء میں باہر شور ہوا تو نہ چلا کہ ماں جی مر گئیں۔ تمام لوگ ماں جی کے گھر ہا گئے۔ انہیں جمل حسین کے بیوی بچے اور اس کا باپ افضل حسین یہی تھا

Noori Nastaliq script by the Monotype Phototypesetter

Figure 9. Machine Printed Urdu Scripts
(Adapted from Noori Nastaliq, Computer se Kitabat, Elite Publishers Ltd., Karachi)

a standardized set of all characters. For languages based on the Greek alphabet, where every letter-form consists of individually placed whole characters, this is sufficient, but not for a script-based language where the sub-word element is often not a whole character. The Monotype system considers ligatures as sub-word elements, and involves the development of a standard set of whole characters and enough ligatures thought sufficient to represent the whole language. The system developed in this thesis, hereafter referred to as Urduscribe, regards not only the whole characters but also all their possible representative shapes, called shoshas, as the basic sub-word elements. This approach required the developing and defining of a new standard set of sub-word elements in order to compose the complete Nastaliq.

Selection of the Standard Character Set. Character sets calligraphed by eight different scribes were compared. Selection criteria were not only based on the elegance and style of the set, but also upon the shosha shapes which would be derived from the set. The exactness and aesthetics of the ligatures composed by such derived shoshas was also important. The character set shown in Figure 10 was selected as the reference to work upon based on these criteria [6].

Principle and Effect of Standardization. Recognizing commonalities in the script made font standardization

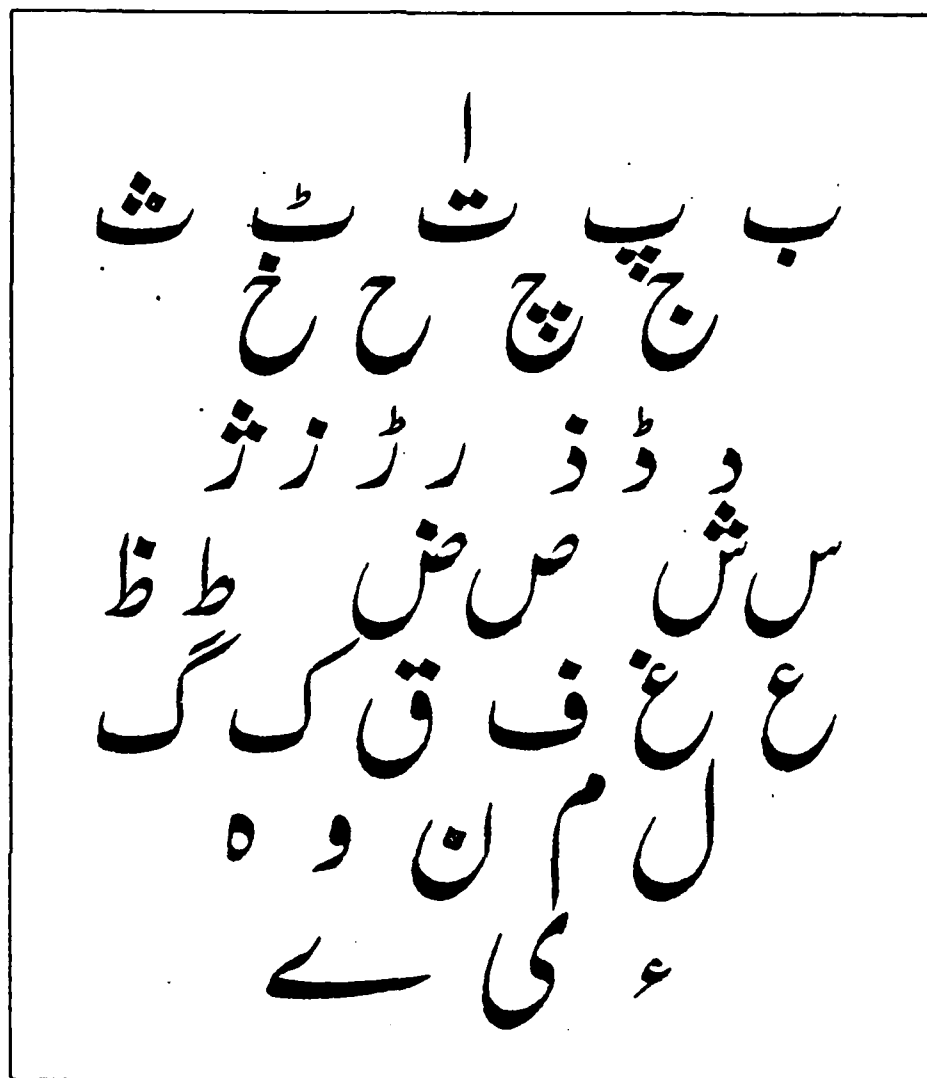


Figure 10. Representative Whole Character Shapes

possible. The benefits realized from the standard font were readily implemented digitization, reduced computer memory usage, and simpler algorithms for composing and displaying Nastaliq. Standardization was accomplished by removing the non-informational differences in those elements of the font that were otherwise similar. At the same time, those differences between shapes that contained information were

enhanced. This principle separates different characters and shoshas in Fourier space, thereby enhancing their recognizability.

Pattern Recognition Considerations

Nukta Evaluation. Nuktas are square diacritic marks standing on their edge. The four possible combinations of nuktas are shown in Figure 13. Nuktas are very important as they distinguish individual characters and shoshas of the same family by their presence, combination, and position. In terms of pattern recognition principles nuktas should have distinct, invariant features so that they can be recognized easily and correctly. In order to accomplish this, three changes were made in nukta shape. First, the nuktas themselves as well as their combinational sets were made symmetrical. This symmetry distinguishes them from those in which nukta position is shifted slightly to be more easily scribed. In addition, the nuktas were slightly enlarged to become easily identifiable. Lastly, their separation with respect to their shoshas or whole characters was increased by a small amount. These changes in the nuktas improve their Fourier domain characteristics so that they demonstrate better identity retention when filtered through a low-pass optical filter. The modifications to the nuktas were carried out after careful observation of their overall effect on the script. To a casual reader these changes would not be noticeably apparent,

yet he would enjoy a comfortable improvement in the font without noticing its cause.

Shoshas in Ligatures. The next place to improve recognizability in the font was to make the shoshas in complex ligatures appear distinct and standard. The scribes have a tendency to squeeze in the shoshas to save space and writing effort, but this diminishes the recognizability and standardization. Due to the rule-driven approach used in Urduscribe, the joints of the shoshas maintain their standardized properties. Special care was exercised in formulating the two character ligature joints to ensure that ample separation exists between the distinct parts of the shoshas, allowing explicit recognition. As the two character ligature formulations have been in many cases extended to create three or more character ligatures, this distinct positioning is also observable in these complex ligatures. Where such higher order ligatures required redefinitions, this underlying principle was kept in sight.

Shosha Derivation from Whole Characters. The 'representative part' of a character signifies its identity. Therefore in most cases the shoshas necessarily retain this part of the whole character, and it can be made consistent in shape for all the shoshas of the same type. Only the connecting portion of shoshas needs to be varied to exactly connect with other shoshas in a ligature. This standardization of the

representative part of shoshas of the same category is also dictated by the pattern recognition principles, which allow shapes to be easily recognized in complex ligatures. Standardization of shapes has also been advantageously utilized in the mechanization of the script.

Digitization of the Font

Decision to Digitize Sub-Word Elements. Once the font had been standardized, digitization of the sub-word elements was considered. The advantages of digital typography are substantial. Sub-word elements can be efficiently coded as discrete and physical properties in any convenient medium. A digitized font can be processed and transmitted as bits of information by a computer, then decoded to reconstitute the original letter-forms at the receiving end. Also the size and shape, as well as subtler qualities of characters, can be readily modified. Unlike analog information, digital signals are highly resistant to noise or degradation introduced during transmission. Moreover, due to the large number of Urdu characters and their shoshas, it was not considered practical to print Urdu Nastaliq using a fully-formed character approach. A dot addressable system of printing, however, offers the needed flexibility for efficiently printing combinations of shoshas and characters. This system requires the digitization of all the individual characters and their shoshas. Printing could then be accomplished by one of two

machines: a high-density dot matrix printer; or a laser printer which smoothens the curves and makes the thickness of character bodies vary gradually, eliminating jagged edges. The same versions of the digitized sub-word elements could also be used to print the script on a graphics terminal if the number of addressable pixels is compatible with the number of printable dots on the printer.

Selection of Matrix Size. In order to digitize the set of sub-word elements, a suitable standard matrix size had to be determined. The height of the matrix was the limiting factor as it determines the ultimate line height. The horizontal width of the matrix would vary from character to character but for any given character would automatically be determined by the fixed aspect ratio. The number of pixels on the screen or dots on the printer increases as the square of the linear resolution of the printing device. Doubling the linear resolution implies a four-fold increase in the amount of information, or the number of bits, that must be transmitted and processed. Although there are computational methods for compressing the data in the bit map of a character or shosha, the general relation between cost and resolution remain valid. Thus the effort was to choose the smallest matrix size that did not compromise letter quality. Also considered was the size of the script which would be printed by the existing hardware for a given matrix size.

As a worst possible case the character Sheen, ش, was selected to help decide the smallest matrix size. Sheen is not only one of the tallest characters but it has the most condensed nukta set; further, its representative part contains maximum detail in the smallest region of space--i.e., the highest information density. Faithful reproduction of the details of the shosha of Sheen by a given dot matrix size would ensure acceptable results elsewhere.

The actual digitization process was initiated by photographing the standardized shapes of every character. Then prints which maintained constant magnification were obtained by careful centering of the negative to avoid projection distortions. The standard magnification ratio was made by having Sheen exactly fill a 5" x 5" square. This character size allowed digitization on commonly available grid sizes and made the pixels large enough to clearly observe the effect of individual pixels on overall character shape.

Special attention was paid to the reproduction of nukta shape, as it is both the smallest element of the script and a very important one. To obtain a symmetrical shape of the Nukta required a minimum of three vertical dots crossed by three horizontal dots. This resolution for the Nukta yields a twenty-four dot high matrix for a complete Sheen of correct proportions. The digitized Sheen thus obtained is shown in Figure 11.

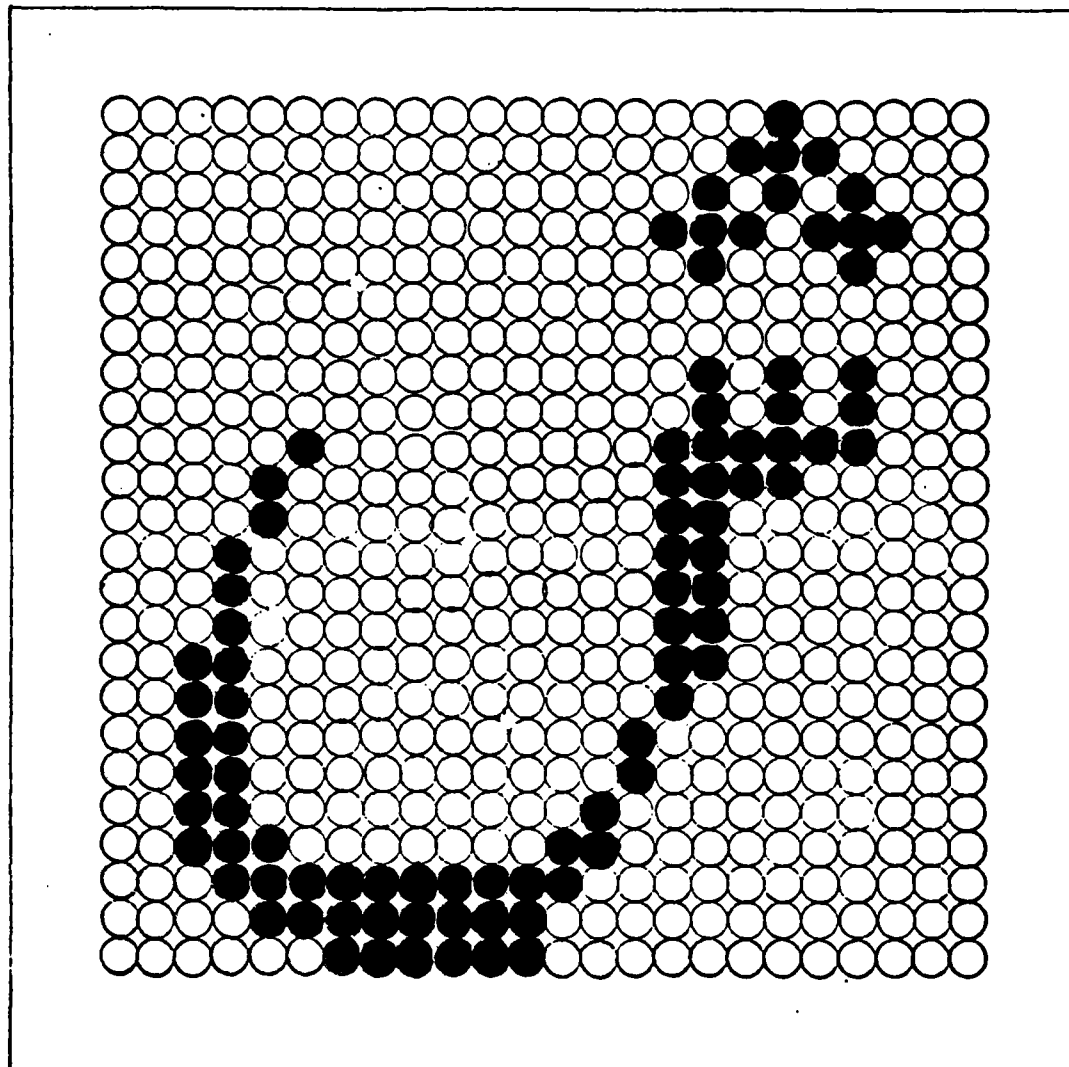


Figure 11. 25 X 25 Matrix Digitization of Sheen

This digitized Sheen was evaluated for trueness by printing it on the screen of a graphics terminal as well as by observing it at a distance through the opposite end of a telescope. The purpose of each of these experiments was to diffuse the high frequency components of the constituent pixels in order to study the trueness of the character shape in isolation. The resulting character shape appeared

satisfactory but the printed version on paper displayed jaggedness of the curves due to the high frequency components of the printed dots. Incorporating this resolution would have made the size of the Sheen 0.14" by the available printer of dot density 180 dots/inch, resulting in a reasonable font size directly usable for conventional Urdu newspaper printing. Unfortunately, the letter quality was unsatisfactory and it was decided to increase the resolution by a step.

Finalization of Matrix Size. Preserving the symmetry of the nukta shape, an increment of a single step in resolution resulted in the representation of the nukta by five dots in both the axes instead of the earlier three dots. This resolution for the nukta translated into a 40 dot high matrix for the whole Sheen. The 180 dot/inch printer could then print the whole Sheen 0.22" tall, making the font size compatible with Urdu typing and the printing of Urdu books. However, the final font size can be easily changed by commonly available magnifying/reducing copying machines.

When the qualification experiments described earlier were used on the new 40-dot matrix, the result was very satisfactory shapes both on the graphics screen as well as on paper. Thus the size of the matrix was finalized and the whole character set and all the shoshas were digitized with this resolution. Figure 12 allows a direct comparison of the digitized and standardized font to the hand-scribed original.

ا ب پ ت ٹ ث
 ج چ ح خ د ڈ ذ
 ر ژ ز س ش ص
 ض ط ظ ع غ ف
 ق ک گ ل م ن
 و ہ ع ی ے
 آ ھ ں

ا ب پ ت ٹ ث
 ج چ ح خ
 د ڈ ذ ر ژ ز
 س ش ص ض ط
 ع غ ف ق ک گ
 ل م ن و ہ
 ع ی ے

Figure 12. Comparison of Digitized Font to Hand-scribed Original

V Mechanization of the Font

Analysis of the Character Set

The Urdu whole character set has groups of elements which look alike. The only difference within a group is the different diacritic group associated with the body of the whole character. These look-alike groups are termed 'character families' or simply 'families.' The individual families are named after the first occurring character of the family. This classification of the character set into families was the first step towards standardization, and is shown in Figure 13.

Urdu whole character bodies can be thought of as being made up of two parts: the shosha which contains the representative part of the character; and the main body of the character which attaches to the shosha. The shosha usually represents the whole character in ligatures. The information contained in the shoshas was made more observable by slightly exaggerating the representative shape, whereas the minor dissimilarities in the body shapes (which contain no additional information) were removed.

Character Body Groups. Another step towards standardization was achieved by recognizing the similarities of the body shapes of certain families. As the primary information is contained in the shoshas, all these similarly shaped

ALIF	آ ا	DAL	د ڈ ذ
BAY	ب پ ت ٹ ث		
JEEM	ج چ ح خ		
RAY	ر ز ژ	SEEN	س ش
SOAD	ص ض	TOYE	ط ظ
AN	ع غ	FAY	ف
KKAF	ق	KAF	ک گ
LAAM	ل	MEEM	م
NOON	ن	VOW	و
HA	ہ	HUMZA	ء
CHOTI YE	ی	BARI YE	ے
DCH	دھ		

Figure 13. Families of Urdu Characters

bodies could be made identical without losing any information or noticeably degrading the font's style.

The clue to the true shape of a character body or a shosha lies in the fact that they are calligraphed by a flat nibbed pen, called the 'Qalm,' which is held at an oblique angle to give the desired varying shape and thickness. When redesigning and analyzing characters and shoshas a similar pen was made and its ductus was traced to obtain realistic relative thicknesses of the sub-word elements.

The bodies of similarly shaped families were analyzed and, with the help of the qualm, six standardized body shapes were designed. These common bodies were so shaped that not only were they useful in isolated characters but could also be used directly in ligatures requiring whole body shapes. The greatest possible part of the original shapes were retained as a set of six bodygroups to maximize standardization. Some characters are so shaped that they had nothing in common with the other character shapes. These characters were treated as single-member families and collectively placed in a seventh bodygroup. The characters comprising the seventh bodygroup include Meem, Ha, Humza, Bari Ye, and DCH.

The portion of a character which is joined with the respective bodygroup to complete the whole character body is called the restbody. In addition to the restbody a diacritic mark is required, in most cases, to be placed with the body of the character to identify the specific member of the family. Usually the diacritic mark is easily distinguishable

as in the case of Nuktas and the Ta (ٹ). In some cases, however, the diacritics are not so apparent, as in Kaf and Gaf. The results of these standardizations and groupings are summarized in Figure 14.

This figure not only formed the fundamental reference for standardization, it also served as an efficient and meaningful system of coding for the character set.

Character Coding. The table shown in Figure 14 was used to determine the first and second place of code for whole characters. The bodygroup of the row determines the first place whereas the diagroup of the column determines the second place. Thus the two digit code reveals the bodygroup and the diagroup of each character. The resulting character coding is listed in Table I.

Standardized Vertical Placement

Unlike the languages derived from the Greek alphabet, Urdu characters, ligatures, and words are placed at different vertical levels in the space of a line. A pattern recognition analysis was carried out to recognize and standardize the patterns of behavior of these letter-forms viz-a-viz their vertical placements. Minor variations were ignored for the sake of standardization and because these minor variations had neither any information content nor desirable aesthetic qualities.

BODY GROUPS	DIACRITIC GROUPS —————											
	A	B	C	D	E	F	G	H	I	J	K	L
1X	ف	ب	غ	ظ								
2X	پ	ت	ث	ق								
3X	ز	ر	ذ									
4X	خ	ج	ح									
5X												
6X												
70												
71												
72												
73												
74												

Figure 13. Classification of Urdu Characters by Bodygroup and Diagraph

Table I
Urdu Whole Character Coding Based on Bodygroup and Diagroup

CHARACTER	CODE	CHARACTER	CODE	CHARACTER	CODE
Alif	50	Ray	31	Kkaf	1F
Bay	2A	Rray	3E	Kaf	2L
Pay	2H	Zay	3B	Gaf	2J
Tay	2F	Yay	3G	Laam	12
Ttay	2D	Seen	10	Meem	70
Say	2G	Sheen	1G	Noon	1B
Jeem	4A	Soad	11	Vow	32
Chey	4H	Zoad	1A	Ha	71
Hay	40	Toye	60	Humza	72
Khey	4B	Zoye	6A	Choti ye	14
Dal	30	An	41	Bari ye	73
Ddal	3D	Ghan	4C	AMZ	5I
Zal	3A	Fay	2B	DCH	74
		NG	13		

It was observed that each whole character appears at the same vertical level when occurring independently. The vertical positioning of all the shoshas in a ligature depends on the position of the last character in the ligature. Normally this last character retains its whole shape. Thus the ligature in most cases could be thought of as being anchored to a vertical level by the whole character at the end of the ligature. This level is the same as when that whole character appears independently. Therefore the vertical levels of all the whole characters in the set had to be defined.

It was observed that characters belonging to the same bodygroup appeared such that the common body preserved its level. Any variation from this rule was inadvertently done by a scribe and was of no consequence. An extensive study of different scripts revealed that the vertical levels of body groups could be standardized to three discrete levels. This result is summarized below:

<u>Baseline</u>	<u>Group Bodies</u>	<u>Vertical Level</u>
1	1, 4, 70	0
2	2, 3, 6, 71, 73, 74	12
3	5, 72	16

The script resulting from this baseline grouping was compared to the scripts written by various scribes and only minor variations were observed.

Positioning of Complex Ligatures Ending in Meem. Meem is one of the tallest characters, consisting of a tall stem bent at the top to support a head shape. What makes it require special consideration in vertical placement is that when it ends a ligature all the shoshas stack on top of it making the whole ligature very tall. If the vertical space of the line was increased to accomodate such tall but infrequent ligatures, it would have resulted in unnecessary space between two lines of script. In order to have compact text, it was decided that for tall ligatures ending with Meem the whole ligature would be moved down to the extent necessary to avoid the top shosha clashing with the upper line of text. However, even if the single thin stem of Meem ran into the text of the lower line the affect would not be noticeably displeasing. Due to the cursive nature of the script, such minor clashes do not produce the jarring effect that they would for a regularly defined and array-positioned characters of a Greek-based language.

EOL (End Of Ligature) Characters

While analyzing the script, ligature formation was closely investigated. A word could be made up of a string of individuals characters, ligatures, or a combination of both. It was found that when certain characters appeared in a ligature the process of joining the shoshas could go no further and the ligature ended there. Thus when such

characters appeared in the beginning of a word or immediately after a ligature, no ligature was formed; rather, the characters appeared independently. These characters were given the name EOLs (End Of Ligature). There are five families of EOLs found in the Urdu script: Alif, Dal, Ray, Vow, and Bari Ye.

Spoilers

The whole shape of a character is usually used when a character appears at the end of a ligature, as opposed to characters appearing elsewhere in a ligature where a shosha shape is appropriate. When a character appearing at the end of a ligature uses other than the normal whole character shape the character is termed a 'Spoiler'. There are four types of such exceptions.

Diacritic Spoiler. When a character appearing at the end of a ligature requires its diacritic to be re-positioned to avoid clashing with shoshas or other diacritics, it is said to have a diacritic spoiler form. Only the character Khay appears in a diacritic spoiler form in two character ligature definitions; its diacritic has two possible positions depending on the preceeding shosha. Many other characters take a diacritic spoiler form when occurring in higher-order ligatures.

Semi-Spoilers. When a character in this category appears at the EOL position, it can have either its original shape or a variant shape that depends on the preceeding shosha in the ligature. Only the characters of the Ray family fall into this category. When a member of the Ray family is preceded by a member of the Seen, Soad, Toye, An, Fay, Kkaf, Meem, Ha, or DCH families, it takes the spoiler form; otherwise it retains its normal EOL shape.

Con-Spoilers. For con-spoilers, the spoiler form of the character is different from its original shape but the shape remains constant regardless of the preceeding shoshas in the ligature. The character families in this category are Dal, An, and Ha.

Noncon Spoilers. Noncon spoilers have many spoiler shapes. The basic shape is that of the whole character, but depending on the preceeding shosha in the ligature various extents of the whole character shape are deleted. The families comprising this group are Choti Ye and Bari Ye.

Noncon Characters

The shape of a character in a ligature depends upon its position in the ligature. When the shape of a shosha changes between its position in the beginning or middle of a ligature, the character is called a 'Noncon', otherwise it is a 'Con'. Noncons may be further divided into Simple Noncons, Bay Noncons, and KAKL Noncons.

Simple Noncons. These are characters which always display noncon properties regardless of neighboring shoshas in the ligature. The simple noncon families are An, Fay, Kkaf, and Ha.

Bay Noncons. Bay noncons are characters which display noncon properties when beginning a ligature and followed by characters from the Seen, Soad, Teye, An, Fay, Kkaf, Vow, and Choti Ye families. The Bay, Noon, and Choti Ye families are Bay noncons.

KAKL Noncons. Characters which change their shape if they begin a ligature and are followed by characters belonging to the Kaf family followed immediately by either Alif or Laam are called KAKL noncons. The families exhibiting KAKL noncon properties are Bay, Noon, and Choti Ye.

Combining Ligatures

In the Urdu language there are various pairs of words which occur frequently in the same combinations. When writing such words there is a tendency among writers to join the ligatures of the words and make them appear as one word. This not only makes the reading and writing of such paired words easy, it also saves space. In the rule-driven approach employed for UrduScribe, a ligature ends when either an EOL is encountered or when an end of word command is given in the form of a space. Therefore if all the characters of the two

words are entered without a space in between, then the complete set of characters is regarded as a single word and displayed likewise. Three examples of such combinations are shown in Figure 15.

Inter-ligature/Inter-word Spacing

The spacing between characters and ligatures is not constant in Urdu. The inter-character space depends upon the two interfacing characters, whereas the inter-ligature spacing depends upon the first character of the succeeding ligature and the last character of the preceeding character. Thus a space definition for all possible combinations of characters and ligatures had to be analyzed. However, after careful consideration the spacings were grouped and standardized in the form of a space dictionary which is used to position characters and ligatures in the text box. Spacing between words can be obtained by moving the cursor a fixed distance with each pressing of the space bar. Thus, spaces isolate individual words. But if a compact script is desired, no spaces need be given between words; then only the usual inter-character and inter-ligature spaces will occur in the script. Absence of distinct spaces between separate words may strike an English reader as very odd and the task of word

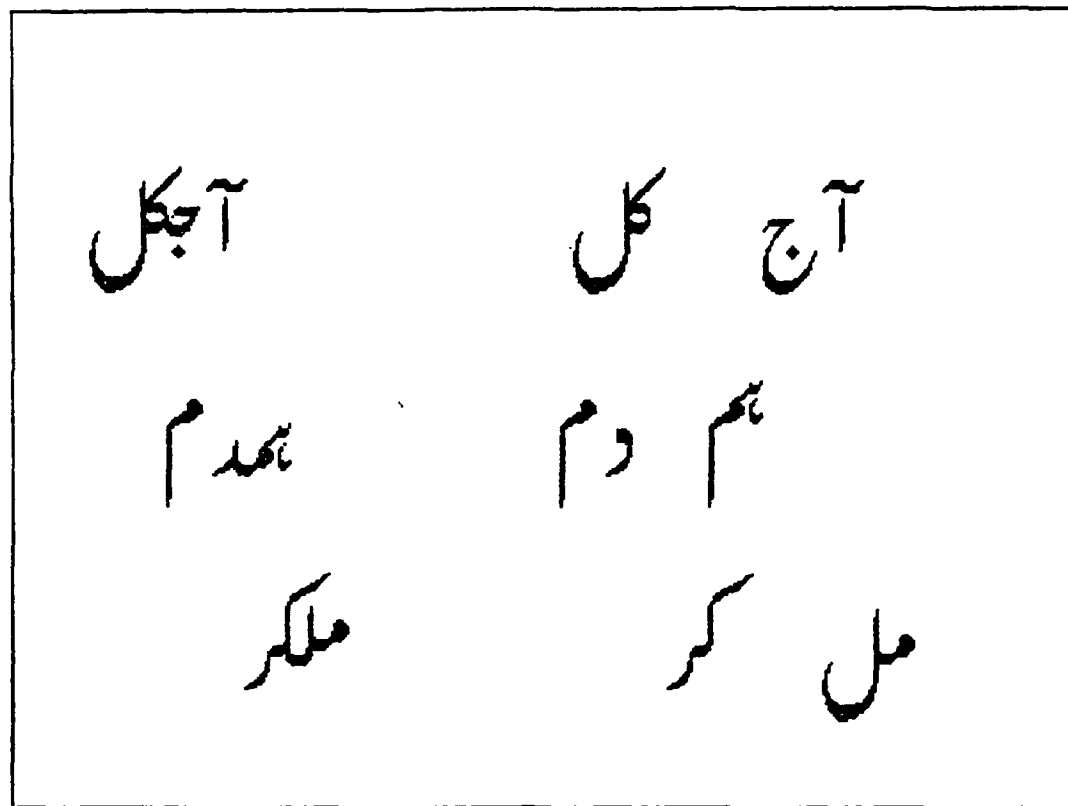


Figure 14. Examples of Commonly Joined Words

recognition as difficult and ambiguous. But due to the cur-
sive nature of the script there is ample flexibility and the
context makes such recognition trivial. A review of the text
samples in Figures 8 and 9, printed both by machines and by
hand, illustrates the absence of distinct spaces between
words. In fact in many cases a character appears closer to a
character or ligature of another word than to other elements
in its own word. Thus depending upon the requirement the
script could be made with distinctly spaced words or be
compacted by omitting such spaces between words. Due to the
special grammatical structure of sentences, missing punctu-
ation marks can easily go unnoticed.

Numeral Handling

Urdu script reads from right to left but the numerals are read from left to right. Numerals are entered into the text buffer, from which text is composed, in the same way--from left to right. In this way, the numerals retain their correct order. When a string of numerals is completed it is moved into the word buffer and then the text box by following the last numeral with a space.

Text Display

Character Buffer. In order to give the operator feedback about the characters typed in, a window was created at the bottom of the screen to display the input string of whole characters. This is the first place a typing error can be identified and corrected. The window can take up to eighteen whole characters at a time; no existing Urdu word contains more than this many characters.

Word Buffer. The word buffer is a window at the bottom right of the screen for the display of newly formed words. The buffer height is greater than that of the character buffer to accomodate tall ligatures. A word in Urdu script is much more easily read in ligature form than as a string of whole characters. Thus it is likely that a mistake not identified when the word appeared as separate whole characters would be found in the word buffer. The composite whole

characters remain on display in the character buffer; thus a word may be corrected before it gets appended to the text box.

Text Box. The text box is a rectangular area on the screen in which final text composition takes place. Text entry starts from the top right of the box and finishes at the bottom left. The vertical spacing of lines of text has been kept 80 pixels to accomodate tall ligatures.

Example of Ligature Formation. Ligatures are formed by adding successive characters according to rules contained in a dictionary. The dictionary specifies the appropriate whole character or shosha shapes and their proper relative positions. An illustrated example of this process is given in some detail in Appendix C.

VI Implementation

Introduction

General. This chapter describes the implementation of the Urdu Nastaliq compositor. The implementation takes the form of a single Pascal program called Newtext. Newtext satisfies three of the four functional requirements specified in Chapter II: text entry, composition, and display. Because of time constraints, text printing was not accomplished, nor were numerals or Aaraabs handled by Newtext.

Newtext was written specifically for use on the Visual Technology VT-550 graphics terminal; the program is therefore highly hardware-dependent. For this reason, the program is analyzed functionally, using flow charts. The flow charts and associated text describing Newtext are ordered in a top-down arrangement. Each procedure included in the program is discussed in detail as it is encountered in the analysis.

Top-Level Description. Newtext may be immediately decomposed into two major elements, Procedure Initialize and Procedure Process_text, as shown in Figure 16. The primary function of Initialize is to load the external files used by Newtext; this process is described in detail in the next section. Initialization is performed before any other function and is performed only once. By contrast, the second major element, Procedure Process_text, executes continuously until the program is terminated. Under Procedure

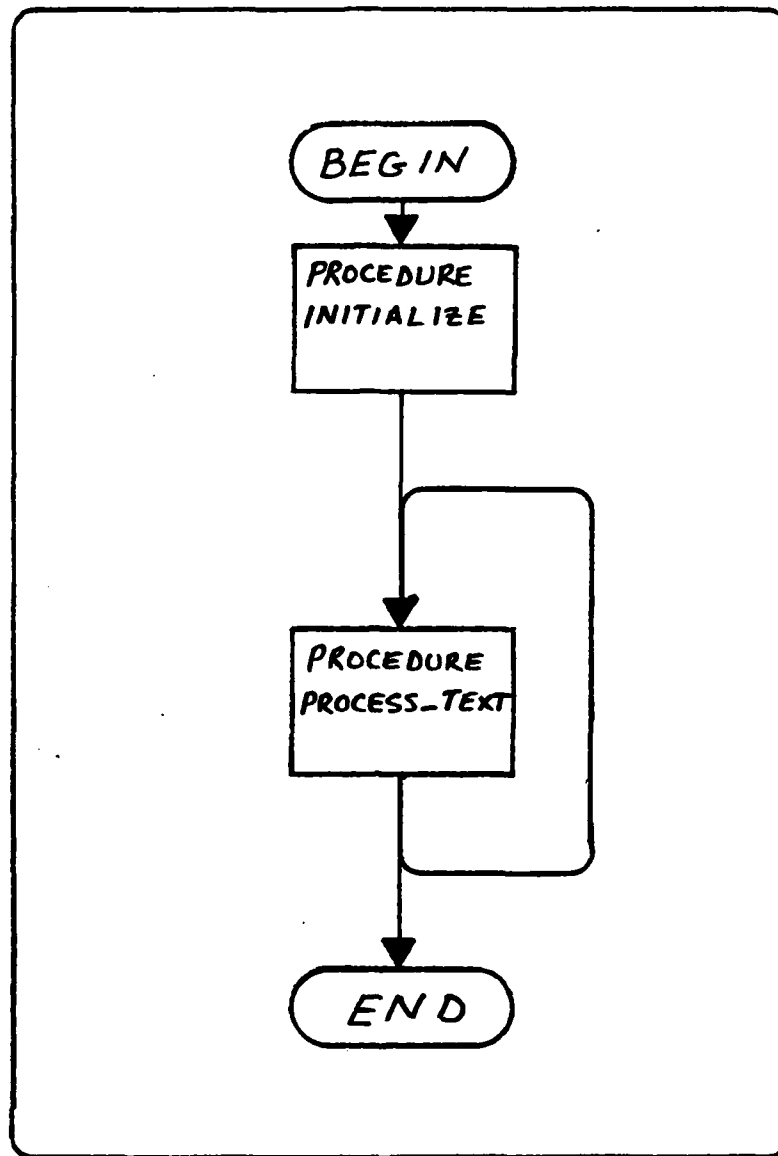


Figure 15. Program Newtext

Process_text, characters are input from the keyboard by an operator. To each key is assigned an Urdu character (Chapter III); as keys are depressed, the corresponding whole character is displayed in its whole form on the terminal's CRT display in a special area called the character buffer. The Newtext ligature-generating mechanisms are activated by

receipt of an operator-inserted space character. When a space is received, Newtext breaks the last input word into one or more ligatures. Each ligature is then displayed, as it is formed, in another special area of the screen referred to as the word buffer. Next, the same ligature is reproduced in a final screen area, the text box. When the entire word has been formed, the typist is given an opportunity to check the word for correctness. The operation of text entry, composition, and display begins anew upon receipt of the next keystroke from the operator. This process continues until the typist terminates the program by depressing the ESC key. As with Procedure Initialize, all the functions mentioned above will be described in greater detail in succeeding sections.

Procedure Initialize

As indicated in Figure 17, initialization is comprised of seven subordinate tasks; each task will be discussed separately. The input to Initialize takes the form of five external files: three files (Bodyfile, Restfile, and Markfile) collectively describing the character font; the file containing the dictionary, or rules, for ligature formation; and the keyboard file which supports the ASCII-to-Urdu transliteration described in Chapter III. Initialization produces data structures as output. The data structures hold the

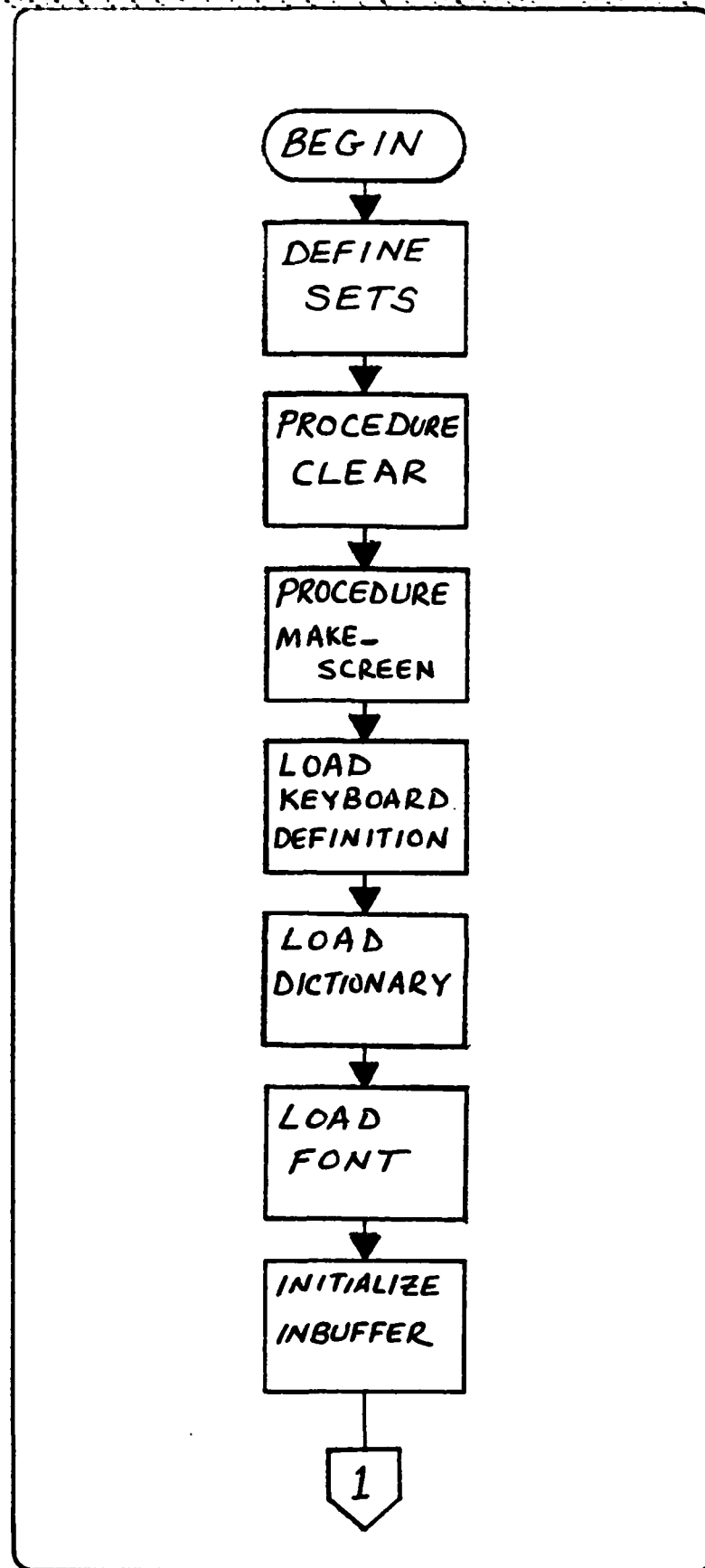


Figure 17. Procedure Initialize

relevant parts of the external files just described as well as certain other important information. Upon completion of initialization, control is passed to Procedure Process_text.

Initialize Inbuffer. Inbuffer, the input character buffer, provides temporary storage for the Urdu characters coming from the keyboard. Inbuffer is an array of characters having dimension 18; this figure is large enough to accommodate all foreseeable valid Urdu words. During initialization, inbuffer is completely filled with a non-printing ASCII character.

Define Sets. After the initialization of Inbuffer is complete, three Pascal-defined set constructs must be prepared. The work required, after defining the sets, is that of filling the sets with their representative elements. The sets to be defined are Valid_char, EOL, and Spoiler.

The requirements specification insists that unassigned keyboard keys be ignored; a major part of the implementation of this function is the Pascal set construct called Valid_char. The characters assigned to Valid_char have a one-to-one correspondence to those keyboard keys having associated Urdu characters. During initialization, the letters which are assigned Urdu characters are loaded into Valid_char. The manner in which Valid_char is used is explained fully in the section on Procedure Process_text.

The significance of End-of-Ligature (EOL) characters is described in Chapter V. The set EOL consists of those letters whose corresponding Urdu characters are EOLs. The assignment of letters to the EOL set is done at this point in the process of initialization.

The third set to be defined is used to specify those Urdu characters which (as described in Chapter V) take on a special shape when they occur at the end of a ligature. Such characters are termed spoilers, and they are detected during later procedures by their membership in the set Spoilers.

Procedure Clear. Clear is the first of several procedures which send hardware-specific control codes to the graphics terminal. As mentioned earlier, only what these procedures do (as opposed to how they do it) will be described. In this case, a single command clears the screen, enters graphics mode, and homes the cursor.

Before proceeding it is helpful to understand the arrangement of the terminal display. The screen has three pre-defined areas: the character and word buffers, and the text box. Figure 18 shows how the display is apportioned to these areas. The character buffer is sized to allow room for up to eighteen full-width whole characters, while the word buffer has room for the ligatures making up any reasonable Urdu word. The text box will allow for the display of six lines of text at one time.

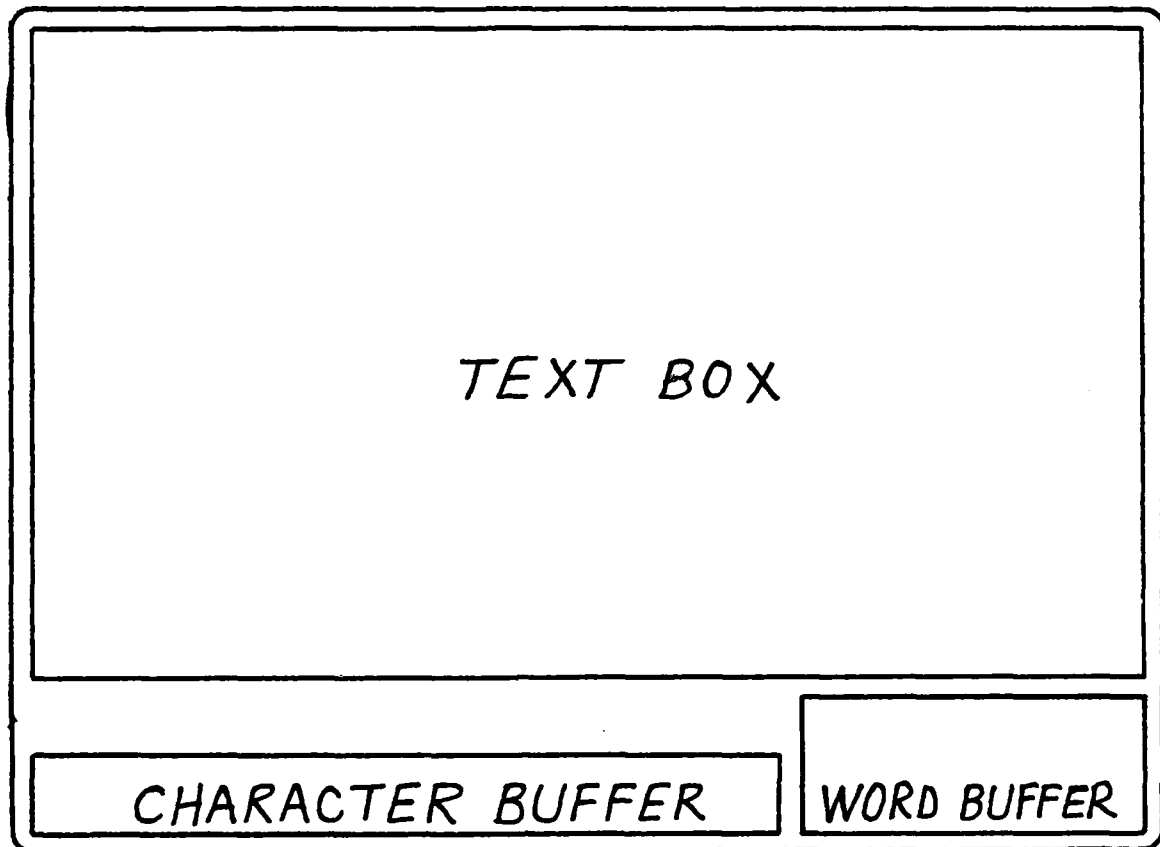


Figure 17. Display Screen Format

Procedure Make screen. This procedure draws the three boxes described above. The first of these is the character buffer, a rectangle of 540 X 55 pixels in the screen's lower left corner. The next box created is the word buffer; it covers an area of 200 X 80 pixels in the lower right corner of the display. The bulk of the screen is taken up by the text box. The size of the text box is 750 X 480 picture elements.

Load Keyboard Definition. Once all three primary sets are defined and the screen display is formatted, it is

necessary to load Newtext's external files. The first file to be loaded is termed Keyboard because it consists of the keyboard definition. The specific ordering of the letters in the file is the basis for the process of transliteration from QWERTY to Urdu. It should be noted that changing the order of characters in Keyboard results in a different layout; consequently, loading Keyboard becomes the means for defining the key-to-Urdu character assignment. During initialization, the contents of Keyboard are loaded into a 42-cell array called Key. Key then holds the ASCII characters corresponding to the forty Urdu characters, the space character, and the carriage return code.

Load Dictionary. The dictionary contains the rules for the formation of all two character ligatures; at this point in initialization, the dictionary is loaded into an array called Darray. Each cell of Darray contains one dictionary entry. The key by which a dictionary entry is found is the integer ID of that element. Another array, called Dhash, is created concurrently with Darray and contains only ID numbers from Darray. There is a one-to-one correspondence between the array indices of Dhash and Darray; that is, when the indices are equal the ID in Dhash is that of the indicated Darray entry. This data structure provides the means for quickly finding a particular dictionary element. The size of Darray and Dhash are specified by a Newtext constant, Numentry, allowing for dictionary expansion if necessary.

Load Font. To load the character font definition actually involves reading three separate external files into corresponding arrays. The files to be loaded are Bodyfile, Restfile, and Markfile; they are loaded into, respectively, Barray, Rarray, Marray. The Newtext constants Numbod, Numrest, and Numdia specify the sizes of the three one-dimensional integer arrays. Each entry of any of the files consists of three hundred integers; however, most entries contain far less than this maximum amount of information so that the files are generally very sparse. To conserve main memory during program execution, the data is compressed by removing the zeros that bring each definition up to its required three hundred integer length. One result is that there are no easily found borders between adjacent definitions to be used in searching. The method used for finding a needed segment is again the hash table, or in this case three tables: Bhash, Rhash, and Mhash. The index of each table is the number of the corresponding segment, while the table entry for that index is the start point of the definition in the appropriate array.

Completion of Procedure Initialize is signalled by the loading of the font. At that point, control of program Newtext is passed to Procedure Process_text.

Procedure Process_text

Text processing, consisting of text entry, ligature generation, and text display, is consolidated under Procedure Process_text. Process_text performs the text entry function directly; further, it invokes the ligature forming and text display procedures.

The structure of Procedure Process_text is best described by the flow chart of Figure 19. At this level, Process_text has two major functions: acquire and display input characters; and invoke ligature generating and display procedures each time a complete word is entered. These two functions are discussed next.

Before text entry begins, three start-up chores local to Process_text are required. First, a pointer is set to the first character position (extreme right side) of the character buffer. Next, the boolean variable Word is reset. Word is true only after the operator has inserted a space, indicating the end of a word; otherwise, as now, it is false. Finally, the character counter, an index to the array Inbuffer, is set to 1.

Once initialized, Process_text awaits a keystroke by the typist. On receipt of a character, called Newchar, Process_text checks to see if the input is an ESC; if it is, the program terminates, otherwise text processing continues.

The next step is to determine if the input was a valid character. This is done by doing a set-membership test on Newchar with the set Valid_char. Invalid input characters are ignored--that is, Process_text loops back and awaits another character. If Newchar is valid, however, another test is performed. In this case, the decision is based on the value of the variable Word. If Word is true, then the previous word has been formatted and displayed; now the character buffer must be cleared of the old word to make room for the new, the character counter reset to one, and Word reset. Control now rejoins the point where it would have been if Word had initially been false. The next step is to insert Newchar into Inbuffer and the input text file Newfile. If Inbuffer isn't full, then the character counter is incremented to indicate the next available cell. Now that Newchar is stored, it gets displayed in its whole-character form in the character buffer. The procedures performing this display function (Key_to_ID, Define_character, and Print_char) will be discussed in a later section. After Newchar is displayed, the character buffer pointer is updated. This amounts to moving the pointer to the left by an amount equal to the width of the character just displayed. If Newchar is neither a space nor a carriage return (CR), then control return to the scan for another Newchar. However, if the current Newchar is either a space or CR then a complete word has been entered. This event signals the word forming operation.

Word formation in Procedure `Process_text` is begun by setting `Word` to true. The `Inbuffer` pointer is then decremented so that it points to the last character of the word rather than to the space or CR just entered. If `Newchar` was a CR, or if the space remaining on the current text box line is less than 30 pixels, then the text box pointers are moved to the right margin of the next lower line. Finally, Procedure `Make_word` is called to cause the formatting and display of the new word. When the formatting and display processes are complete, `Process_text` will once more take over, waiting for a new input character from the operator.

Procedure Make word. The function of Procedure `Make_word` is to compose and print a single word, one ligature at a time. To accomplish this, `Make_word` begins by performing three preparatory actions. First, the word buffer pointers are reset to the right margin of the box; next, the word buffer is cleared; and finally, the first cell of array `Inbuffer` is checked for a space. If the first cell is a space, then the word has already been formatted and displayed, and control reverts to `Process_text`. Figure 20 makes this process somewhat clearer. If, however, the first cell does not contain a space, then Procedure `Make_ligature` is called. `Make_ligature` composes, then causes to be displayed, the first ligature of the current word. `Inbuffer` is checked again, and the process is repeated until the entire word is done. At that point, the first cell of `Inbuffer` would

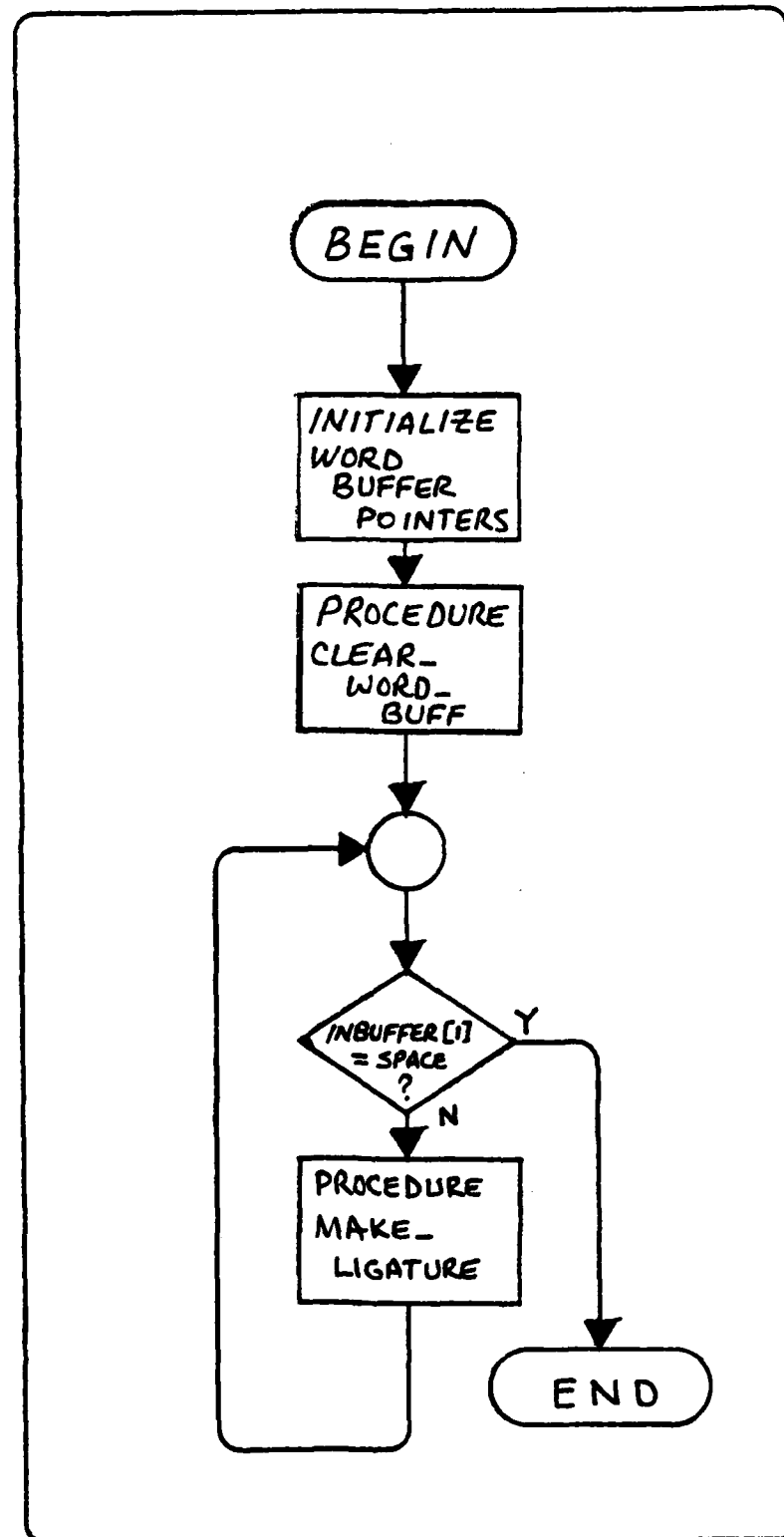


Figure 20. Procedure Make_word

contain the original operator-inserted space that resulted in Make_word being called, and Process_text takes over.

Procedure Clear word buf. This procedure simply rewrites the word buffer on the display screen by turning off all the pixels within the word buffer.

Procedure Make ligature. Procedure Make_ligature is responsible for breaking words into ligatures, then having those ligatures displayed, first in the word buffer and subsequently in the text box. Upon entering Make_ligature, the first character in Inbuffer is tested for set-membership in set EOL (see Chapter V). As is indicated in Figure 21, if the first character is an EOL, it is printed immediately; otherwise, Procedure Find_EOL is called to scan Inbuffer for the first EOL character and hence the end of the first ligature. Make_ligature then invokes Procedure Print_char twice--once to put the ligature in the word buffer, and once to place the new ligature in the text box. Finally, Inbuffer is shifted to remove the characters of the just-displayed ligature. Control now reverts to Make_word.

Procedure Find EOL. As already mentioned, Find_EOL identifies the first EOL character, if any, in Inbuffer. Find_EOL returns an integer value, Liglength, which is both a pointer to the EOL character and an indicator of the ligature's length. It should be noted that the last character in a word is an EOL by definition.

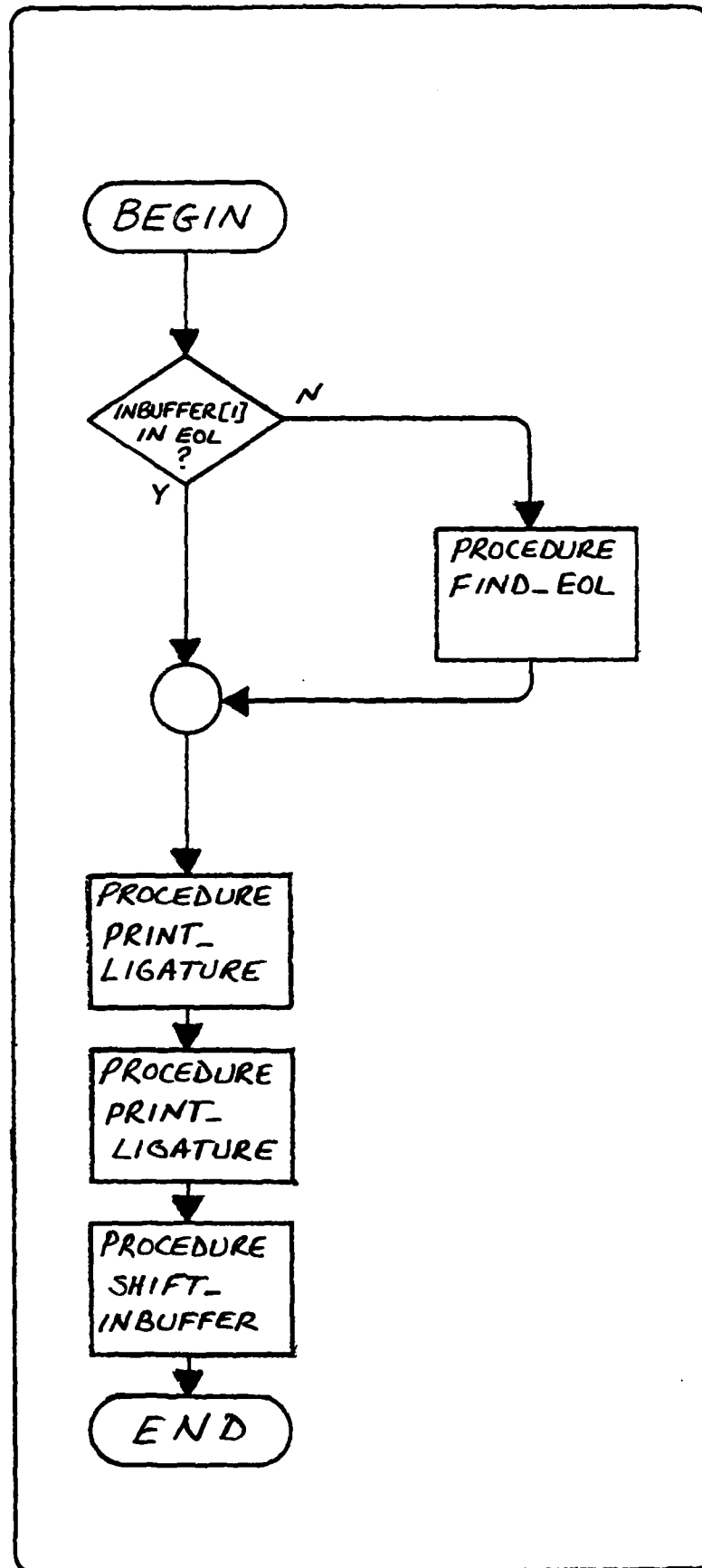


Figure 21. Procedure Make_ligature

Procedure Shift Inbuffer. When invoked, Procedure Shift_Inbuffer shifts the Inbuffer to the right to remove the first ligature. The number of shifts is equal to Liglength, the number of characters in the ligature.

Procedure Print ligature. The real heart of program Newtext is Procedure Print_ligature. It is here that shoshas from the character font are joined and displayed according to the rules of ligature formation found in the dictionary. Upon entering Print_ligature, as shown in Figure 22, two possibilities arise: the ligature to be printed is either a single character or a multiple character ligature. If a single character ligature, the Procedure Key_to_ID is called to get the character ID; then the whole character definition is retrieved by Procedure Define_char, and the character is displayed by Procedure Print_char. On the other hand, if the ligature consists of more than one character, then there are again two possibilities: either the EOL character is a spoiler (Chapter V) or it retains its whole character shape. The decision is based on set membership of the EOL in set Spoiler. In either case, the proper character ID is obtained and the appropriate character definition is read from the dictionary. Similarly, the shosha definitions for the remaining characters of the ligature are obtained and placed in the array Charay. When the process is complete, Charay will contain one definition for each character in the ligature. Once Charay is filled, the ligature width must be determined

so that the pointers in the text box and word buffer can be moved accordingly. Ligature width is computed with the help of Procedure Get_RP_SP. After the width is computed and the pointers moved, the start points of the ligature is found by a second pass through Get_RP_SP. If the ligature width exceeds the available space on the current text box line, then the text box pointers are moved to the beginning of the next line. Finally a loop is entered in which the contents of Charay are passed by Procedure Pass_char to Procedure Print_char for printing. This loop is exited after the ligature's EOL character has been displayed.

Procedure Key to ID. This Procedure successively compares the current character to the contents of the array Key. When a match occurs, the index of array Key is the ID of the given character.

Procedure Define character. In Define_character, the character ID is used to fetch the proper dictionary entry from Darray. The table Dhash is used in this search as explained in an earlier section.

Procedure Get RP SP. Get_RP_SP uses information from Charay to determine the start point for displaying the ligature. This same information can be used for finding the overall ligature width.

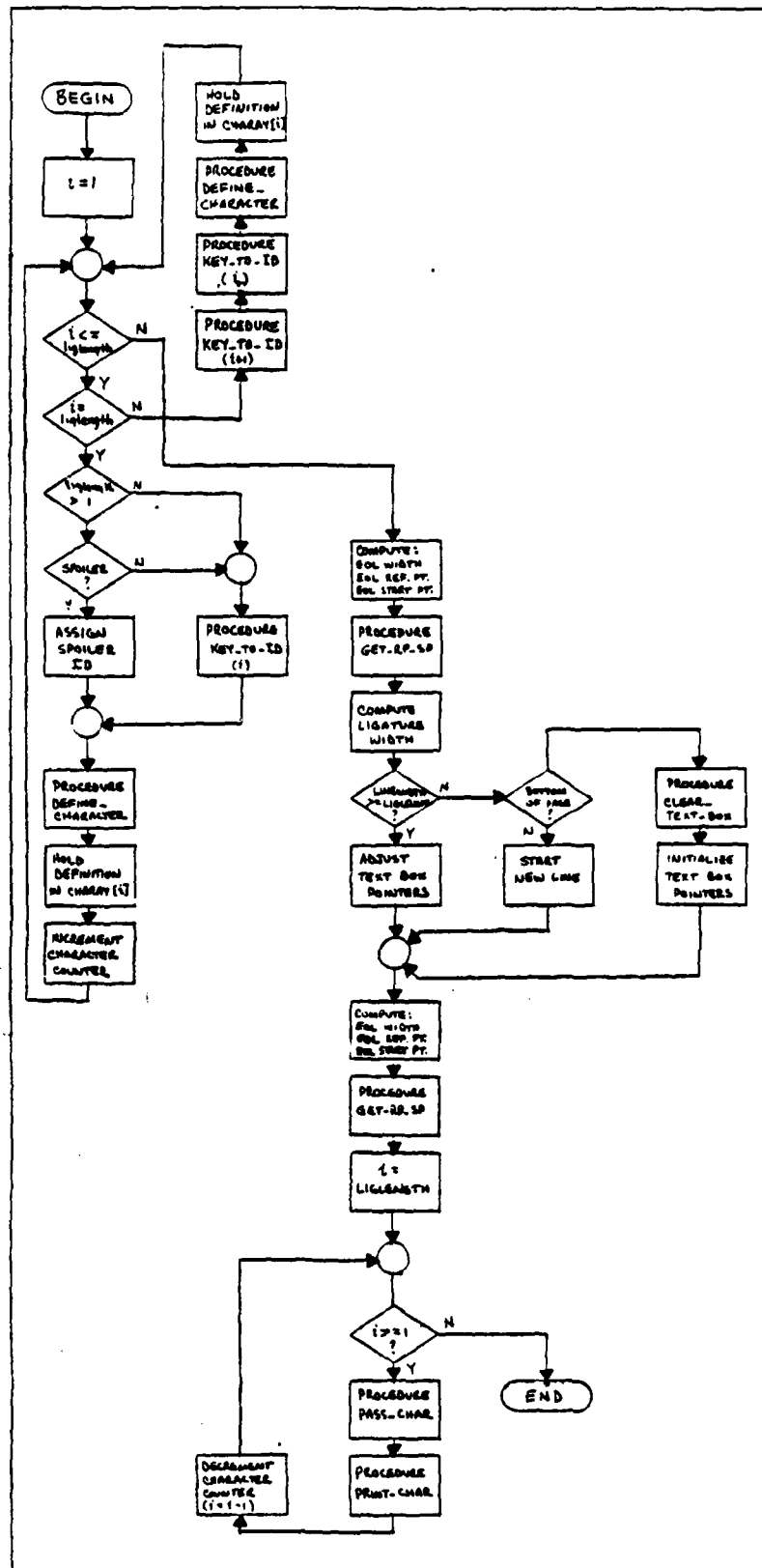


Figure 22. Procedure Print_ligature

Procedure Pass_char. In this Procedure a single character definition is passed from Charay to Print_char for display.

Procedure Print_char. The functions and elements of Procedure Print_char are detailed in Figure 23. A simple decomposition of Print_char shows it to be made up of three similar segments. In the first segment, the body group number, passed by Print_ligature from the character's dictionary entry, is used to find the corresponding number in Bhash, the bodygroup hash table. The hash table entry points to the beginning of the body definition. The definition consists of integers which denote pixel addresses, and each definition contains from 1 to 300 such integers. The exact number of integers need not be known in advance, since the last valid address of each body is followed by a -10 entry. As each address is gathered, Procedure Conv_coord is called to perform a conversion of the address to a form usable by the graphics terminal. After the last pixel of the body is displayed, a test for a corresponding restgroup is made. While every shosha and whole character must have a bodygroup, some do not require a restgroup. If the dictionary entry for the restgroup is a zero, then there is no restgroup; if non-zero, then a procedure identical to that just described is followed to display the restgroup. Whether a restgroup is defined for a character or not, a similar test is made to determine if the character requires a diacritic mark.

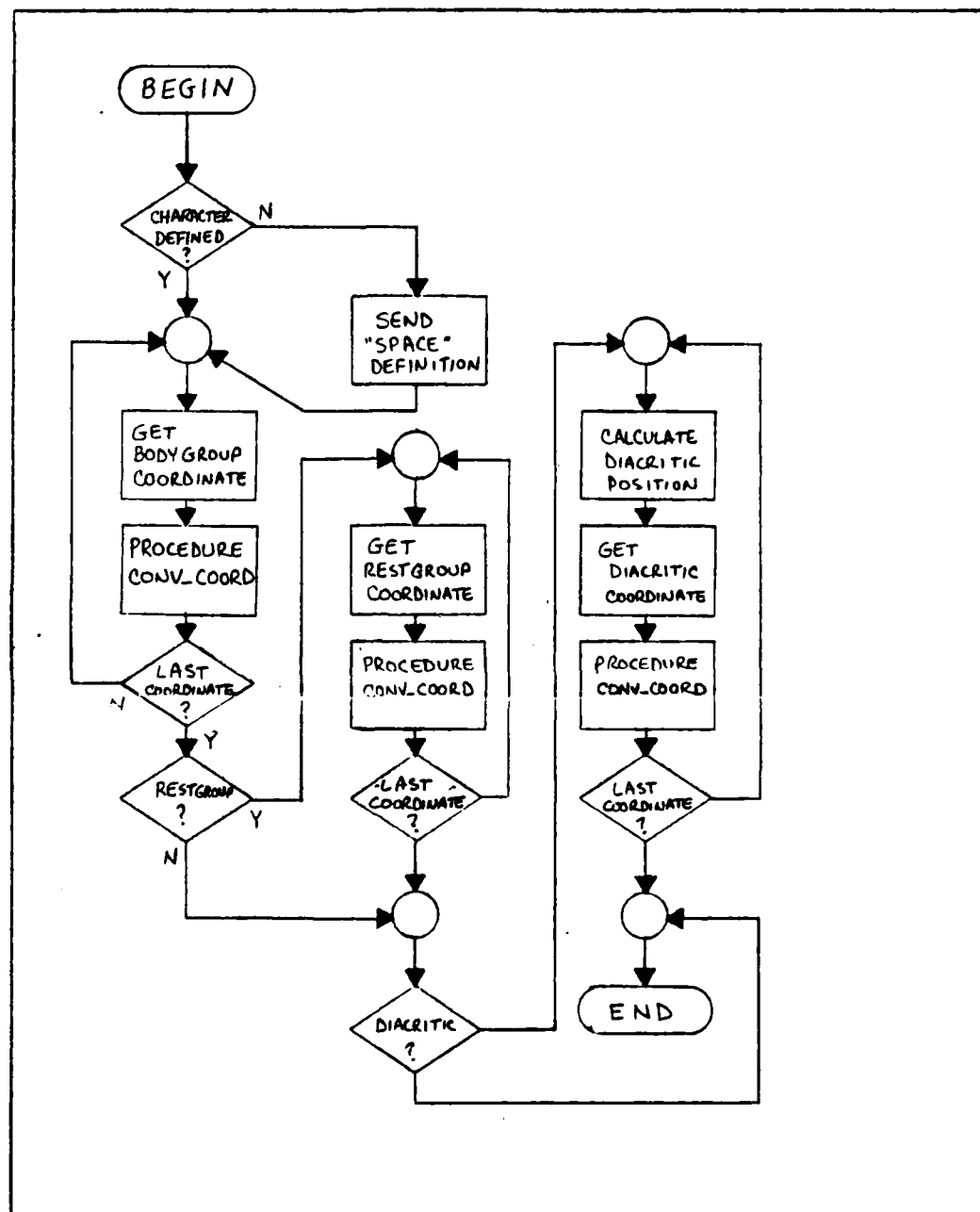


Figure 23. Procedure Print_char

VII Testing and Analysis

This chapter describes the testing and analysis of the Urdu Nastaliq compositor implemented in program Newtext. Four test areas were defined by the acceptance criteria of Chapter II: text entry, composition, display, and printing. Tests were developed to measure Newtext's compliance with each criterion in the four test areas. The tests, and corresponding results, are presented and analyzed here.

Testing

Text Entry. The first text entry specification requires that all necessary Urdu elements be enterable from the keyboard. While the current version of Newtext provides for the entry of the set of whole characters, it does not presently accept numerals, Araabs, or punctuation.

The second part of the specification insists that the mapping of Urdu elements to keyboard keys be efficient. As explained in Chapter III, the mapping of Urdu whole characters to the standard QWERTY keyboard is very efficient in terms of matching character frequency to finger agility. Test data on typing speed was not collected for two reasons. First, a masked keyboard showing Urdu characters rather than English characters was not available for long-term use. Second, there are no typists, trained on the new keyboard, to

be used for data collection. Even if figures on typing speed had been taken, they would only have been significant in comparison to similar data for other existing Urdu keyboard layouts; such information is not readily available.

A test was run in which every keyboard key except BREAK and DELETE was pressed during Newtext execution. The results showed that Newtext ignored all entries but those explicitly defined by the program. Also per the specification, Newtext created a file of ASCII characters corresponding to the valid inputs from the keyboard. This file is stored in secondary storage after Newtext is terminated to permit later printing or editing.

Text Composition. To completely satisfy the text composition specification, six criteria had to be met. Of the six criteria, five were completely fulfilled while the sixth requirement was partly met. As per the specification, the operator insertion of a space starts the word composing process. Single letters followed by spaces are printed as whole characters. Ligatures are formed automatically, as part of word composition, using the rules contained in the dictionary. Characters and ligatures are placed at the proper relative vertical position as part of ligature formation. Horizontal spacing between characters, ligatures and words is maintained; however, the spacing is wasteful of space.

AD-A138 067

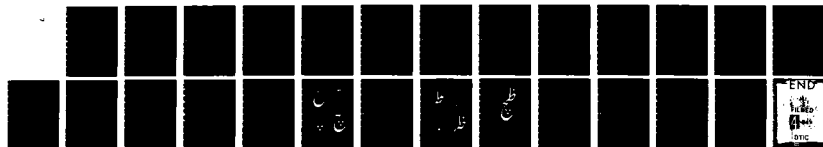
A SCHEME FOR THE COMPUTERIZED COMPOSITION OF URDU
NASTALIQ(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB
OH SCHOOL OF ENGINEERING A H KIZILBASH ET AL.
06 DEC 83 AFIT/GE0/EE/83D-2

2/2

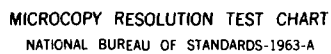
UNCLASSIFIED

F/G 9/2

NL



END
6-44
100



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Text Display. The criteria for text display covers six areas of display performance. Four of six criteria were judged to be satisfied; the two remaining areas were not implemented in the current version of Newtext. The program displays whole characters immediately after they are entered. Further, text is displayed from right to left and top to bottom. Finally, word wrap is automatically performed by Newtext on ligature boundaries. Newtext does not currently handle numerals, however, nor can it claim to provide a one-to-one character correspondence between displayed and printed characters. This last is because the printing of the text on the Toshiba high-density printer was not accomplished.

Text Printing. As mentioned above, text printing on the high-density printer was not possible due to lack of time. Printouts were obtained by taking screen dumps from the VT-550 to a Data South 180 printer. While these printouts were satisfactory for showing that ligature generation worked, they were unsatisfactory in terms of the criteria of Chapter II.

Analysis

Whereas the testing done earlier was to measure the compositor's adherence to a set of acceptance to a set of format-based criteria, this analysis focuses primarily on UrduScribe's language handling capability, character font, complex ligature formation, and Urdu text composition.

Analysis of Language Handling. Not all features of Urdu Nastaliq were implemented because of time constraints. The approach chosen, therefore, was to solve the general text composition and display problem first; then exceptions and special cases were handled on a priority basis. Based on this idea, the keyboard and font designs were completed first. Program Newtext was written to construct arbitrarily long ligatures by repeated application of the two-character rules. This capability was considered the minimum for the project to be deemed a success.

Enhancements were made as time permitted, including coding of the logic to correctly handle spoiler characters as EOLs. Several significant tasks remain to be accomplished in text composition and display; these features and the logic for their delayed implementation are enumerated below.

Analysis of Character Font. Close inspection of print-outs of the standardized character set prompts several observations. First, the aesthetic qualities of Urdu Nastaliq have survived the standardization attempt as can be seen in Figure 24. The construction of characters from common templates has ensured that all members having the same bodygroup maintain their proper relative sizes and adhere to the appropriate baselines. The diacritics, by being made more distinct and being moved away from the group body, are more

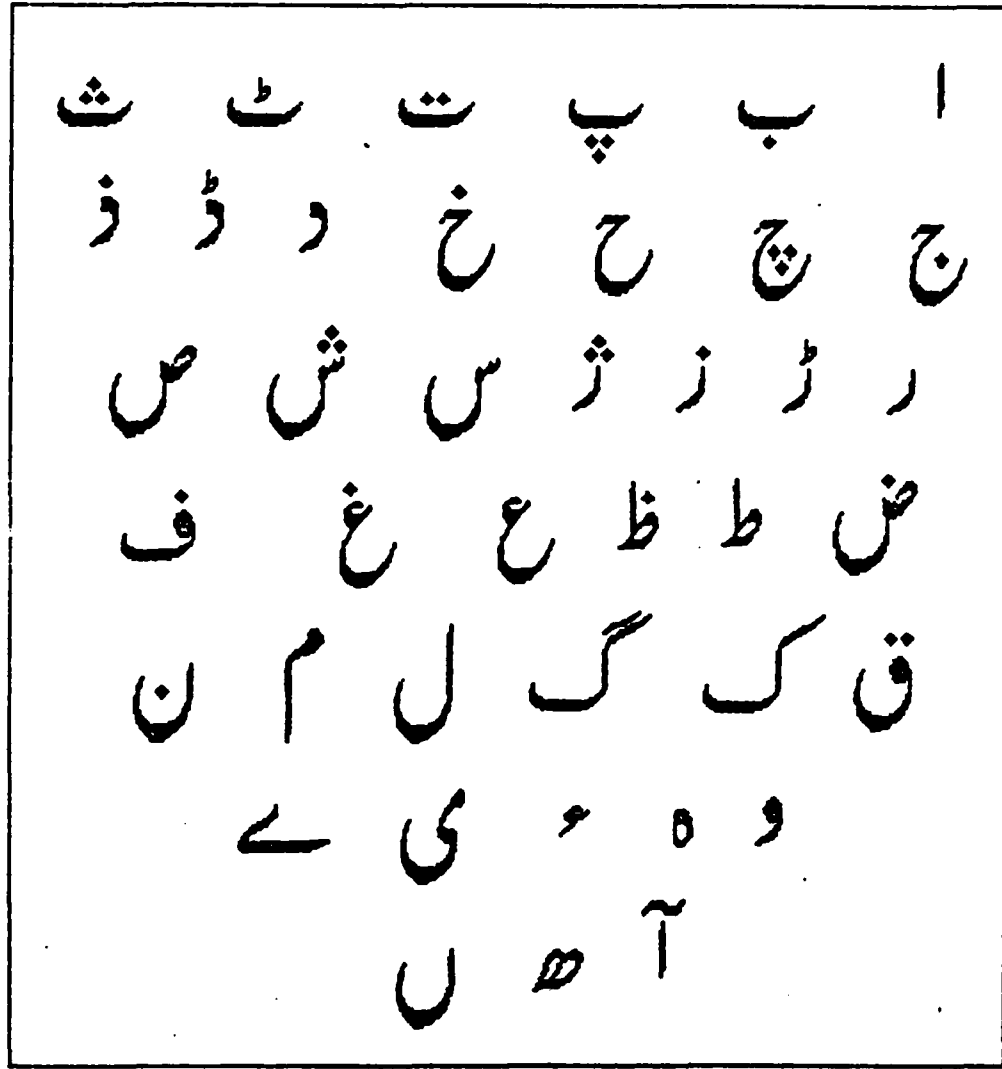


Figure 23. Urduscribe Standard Whole Character Set

easily seen and recognized. Finally, the font is extremely readable in spite of a small amount of jaggedness derived from the digitization process. This effect can be minimized by increasing the dot density of the printer or employing a printer that smears the dots slightly.

Analysis of Ligature Formation. The greatest success of this project was the faithful reproduction of complex ligatures in Urdu Nastaliq. Two features of the technique employed are especially pleasing. First, the joints between shoshas in ligatures are smooth; and second, the shoshas are connected at exactly the proper relative points. An example of such a complex construction is the 15 character ligature shown properly formatted in Figure 25.

Analysis of Urdu Text Composition. There are two noteworthy observations on overall text composition. The first observation concerns line height. Initially line height was set at twice the average whole character height; this appears to have been overly conservative. An approximately 20% reduction in line height would increase text density by the same amount without causing inter-line collisions. An inspection of the sentences in Figure 26 shows how much "wasted" space exists between lines with the existing line height.

The second remark on text composition is that complex ligatures tend to be compressed by scribes to save time and effort. This compression may take the form of reduced inter-shosha spacing or manipulation of the basic shosha size and/or shape. By contrast, the size, shape, and spacing of shoshas composed by Urduscribe remains constant even for complex ligatures.

بلگسجکستط

بل گسج کآم صط طه ح گآم

Figure 25. An Example of Complex Ligature Formation

گھوڑا گھاس کھاتا ہے
ماں نے بچوں کو ماتھے پر چوما
عشق نہ پوچھے ذات

Figure 26. Sample Sentences Printed by Urduscribe

VIII Conclusions

On the basis of results obtained in developing a means for the computerized composition of Urdu Nastaliq, the following conclusions are drawn:

1. The rule based approach to ligature generation offers the greatest flexibility and efficiency, and represents a complete solution to ligature formation.

2. Standardization of the sub-word elements greatly enhanced the pattern recognition qualities of the font. This standardization was accomplished by making common elements identical and by enhancing the differences of the character parts bearing information.

3. The script was efficiently composed and displayed by the digitization of sub-word elements.

4. The keyboard layout design is based on a character-to-key mapping obtained by matching character frequency with finger agility, using pattern recognition principles in character pairing, and rationalizing of the finger work loads.

5. The use of a dictionary of ligature-forming rules enabled the efficient construction of arbitrarily complex ligatures by repeated application of simpler rules. Exceptions to the general rules require only additional dictionary entries.

6. Program Newtext tends to become I/O bound during execution because of the tremendous amount of graphic data required to display the script.

IX Recommendations

Based on observations made during the design and analysis of the compositor, the following recommendations are proposed:

1. Rules for printing Noncons in third and higher order ligatures should be included in the rules dictionary and implemented in Newtext.
2. Araabs and punctuation should be assigned to vacant keys using criteria described in chapter III.
3. Numerals, Araabs, and punctuation should be incorporated into the Newtext software.
4. A scheme to correctly position complex ligatures ending with Meem needs to be devised.
5. A separate dictionary for determining interligature horizontal spacings should be developed.
6. Line height should be reduced to increase text density while still preventing interline collisions.
7. Newtext should be expanded to allow printing of Urdu text on the Toshiba P-1350 printer.
8. A 25x25 dot matrix font for use on the display screen should be developed to make the linewidth on the VT-550 terminal equal to the linewidth on the P-1350 printer.

This would also reduce by half the time spent in sending characters to the display screen.

9. The word Allah should be drawn, digitized, defined in an appropriate dictionary entry, and assigned a vacant key.

10. A keyboard should be masked per Figure 7 to permit data collection on typing speed for the Durb-Kizil keyboard.

11. The physiologically optimized keyboard described in Appendix B should be built and tested.

12. Program Newtext should be transported to a dedicated microcomputer. Text input should be interrupt-driven and have priority over display functions. The microcomputer should have a dedicated graphics terminal driven by the computer's main memory.

Bibliography

1. Naqvi, Jamil A. "Nastaliq--the Elegant Urdu Script--Its Origin and Progress from the Sixth Century to the Present Day," The Monotype Recorder New Series, Number 3, October 1981.
2. "Noori Nastaliq," Computer say Kitabat, Elite Publishers, Ltd., Karachi, 1982.
3. Advertisement appearing in The Daily Jung, Karachi, 25 July, 1983.
4. Adler, Michael H. The Writing Machine. London: George Allen and Unwin, 1973.
5. Lemmons, Phil. "A Short History of the Keyboard," BYTE New York: McGraw-Hill. (November 1982).
6. Sirajuddin, Zafar. Phool Aur alyan, Pehla Hissa. Feroz Sons, Ltd., Lahore. (undated).

Appendix A

Alternative Approach to Keyboard Assignment

Although the total number of independent characters was concluded to be 40, an alternative approach was conceived which could reduce the total number of character keys from 40 to only 28. The idea was to make use of what is called the family relationship between most of the characters. A family is composed of characters having the same shape with only the diacritic marks and their positions being different. The whole character set was divided into families as shown in Figure 13.

In this manner the character set can be reduced to a total of 20 families and 8 diacritic marks. Thus a total of 28 keys, one for each body shape and diacritic mark, could have sufficed to produce the whole character set. This relatively small keyboard appeared to be a costly trade-off in terms of typing effort, however, because a total of 23 of 40 characters have diacritic marks. This would require the use of two keys for those characters having diacritics. The result would be extra time and effort for the typist. Thus this idea was shelved, and the regular mode of assigning a single key to a whole character was adopted.

Appendix B

The Physiologically Optimized Keyboard

Ever since the invention of the typing machine, keyboards of widely diversified designs have come into existence and faded away with time. There were circular keyboards, hemispherical keyboards with keys protruding like the spines of a porcupine; then there were the straight-row keyboards with various number of rows. Somehow the last category of keyboards dominated the scene and have gotten themselves established. However, the straight-row keyboards are not suitable to the human physiology. It is anatomically unnatural for the arms to be clasped to the trunk and then for the hands to be held straight in front and parallel to each other. The more natural position is to let the arms relax and take their own position with the trunk, angling inwards. This allows the wrist joints to be straight so that the hands assume an oblique position in front. When the fingers are naturally bent the finger tips constitute an arc and not a straight line due to their different sizes and the way the hand is made. Instead of straight rows, the keys should emulate this relaxed position of the finger tips. Thus for each hand the keys should be located along an arc curving inwards as shown in Figure 27.

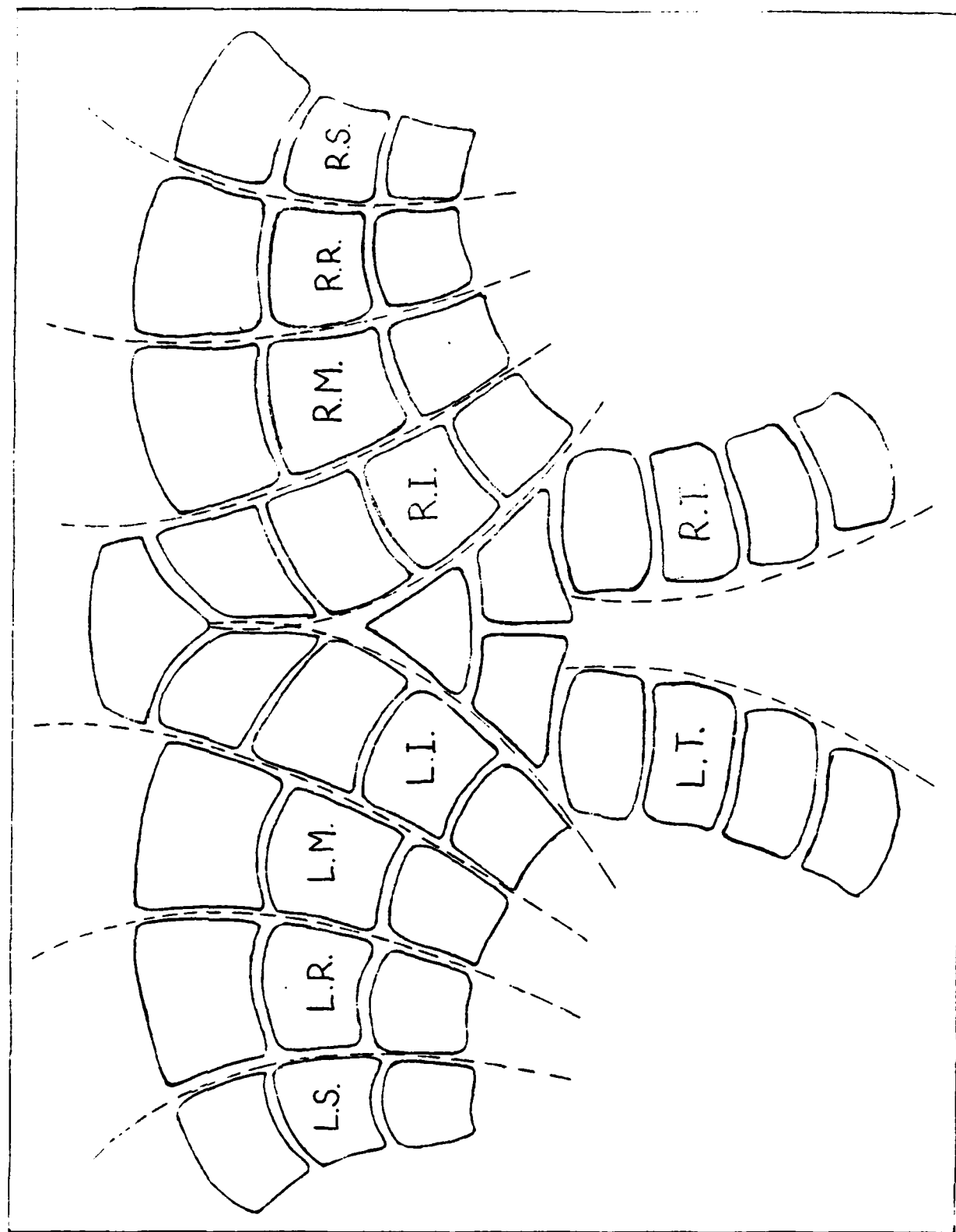


Figure 27. The Physiologically Optimized Keyboard



The way the fingers get positioned when the hand is opened differs from person to person due to slight anatomical differences. Thus the average area desired for the positioning of the outer circular row of keys should be increased. Due to the diverging circular arrangement, the keys of the outer rows would automatically have more space which can be utilized to make those keys bigger so that people with differently sized fingers can naturally reach the correct key. On the inner row the fingers tend to be at the same spot and the variance is reduced. There the keys can be made smaller, which is also dictated by the lesser space available there.

On the conventional keyboard the thumbs are lolling on the space bar and get to do much less work than they are capable of performing. In fact, the thumb is the strongest finger and can be quickly and easily moved in an axis lateral to its length. Each thumb can easily recognize as many as four distinct key positions along its axis of movement. Therefore together the thumbs can work eight keys, taking a substantial load from the fingers, making the job of typing faster, easier, and less tiring. Moreover the function keys which presently have to be accommodated at the extreme ends of the keyboard and operated by the weakest fingers can now be placed in the middle of the keyboard between the outermost rows. This makes them available to the efficient fingers of both hands.

The exact and actual layout of such a keyboard requires extensive collection of statistical data on human hands. However Figure 27 is an approximate layout. Its effectiveness could be proven by conducting competitions with the conventional keyboard under realistically balanced conditions. All this work was beyond the scope of the present project and therefore has been recommended for further investigation.

Appendix C

Illustrated Example of Ligature Formation

The purpose of this example is to illustrate by example the formation of a two-character ligature. The ligature to be composed consists of Chey () and Zoye ().

As explained in Chapter V, Chey is the EOL (End of Ligature) character, so it is displayed first; it forms the base on which the rest of the ligature is built. Since Chey is not a spoiler, it retains its whole character shape as an EOL. Referring to the rule dictionary, it is found that Chey is composed of a bodygroup, restgroup, and diagroup, which are shown in Figure 28(a), (b), and (c). The square in the lower right-hand corner of each element of Chey is the "start point" for drawing the element. In constructing a character, the start points of the bodygroup and restgroup always overlap; the start point of the diacritics usually differ by an (x,y) offset, called the diaposition, which varies from character to character. In this case the diacritic start point is offset from the bodygroup/restgroup start point by x=5 and y=9 pixels. Overlaying the bodygroup, restgroup, and diagroup (with proper offset) results in the whole character form of Chey, as shown in Figure 28(d).

Next, the proper shosha form of Zoye must be chosen. Referring again to the rule dictionary, the shosha is identified. The shosha, like the whole character, is

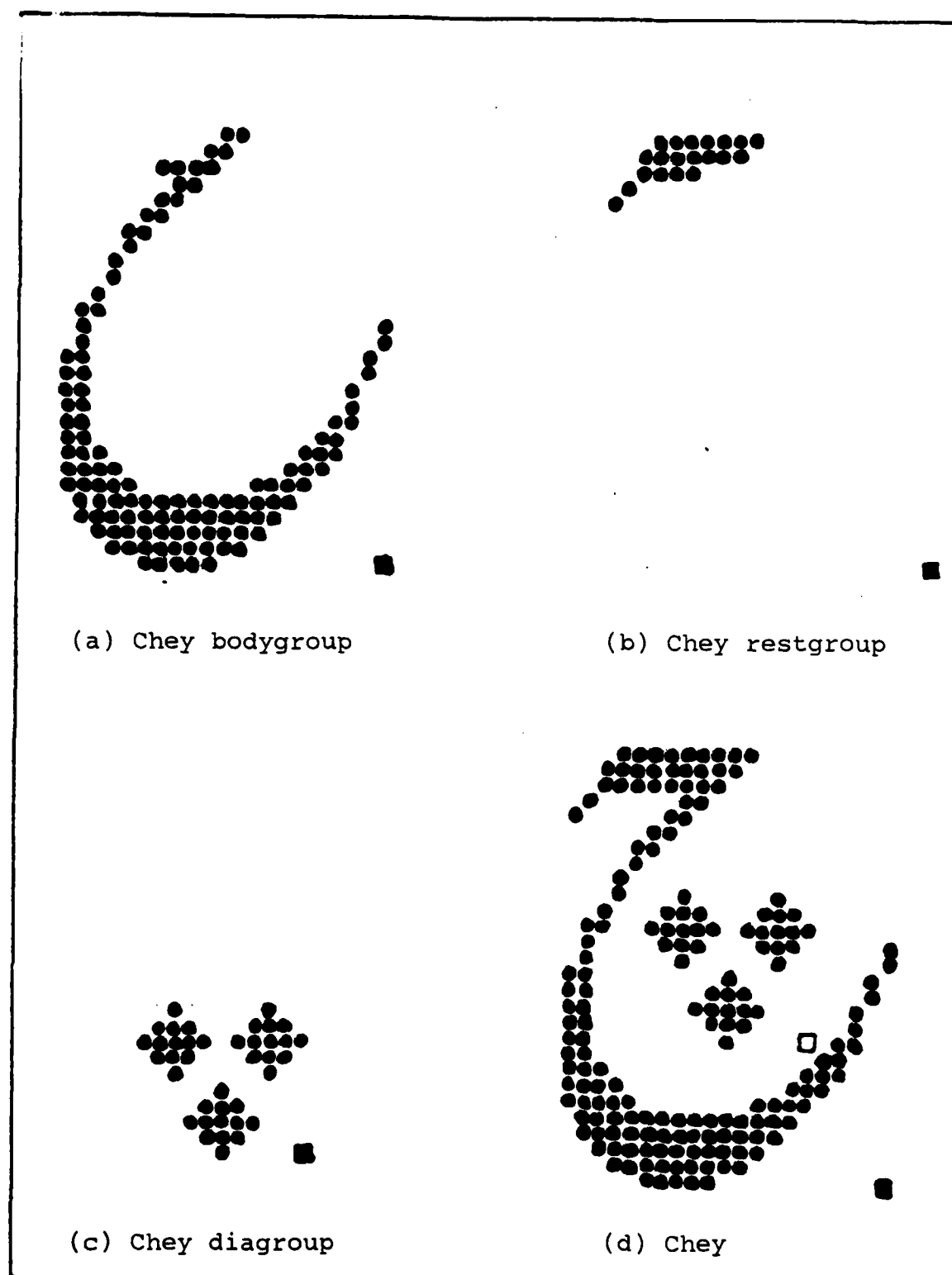


Figure 28. Elements of the Whole Character Form of Chey

composed of a bodygroup, restgroup, and diagroup. These elements are reproduced in Figure 29(a), (b), and (c). Joining a shosha with a whole character or another shosha requires an offset which places the successor shosha relative to its predecessor; this information, too, is contained in the dictionary entry. Combining the shosha bodygroup, restgroup, and diagroup yields the shosha of Figure 29(d). Using the offset from the dictionary, the Zoye shosha is joined with the whole character form of Chey, resulting in the two character ligature in Figure 30.

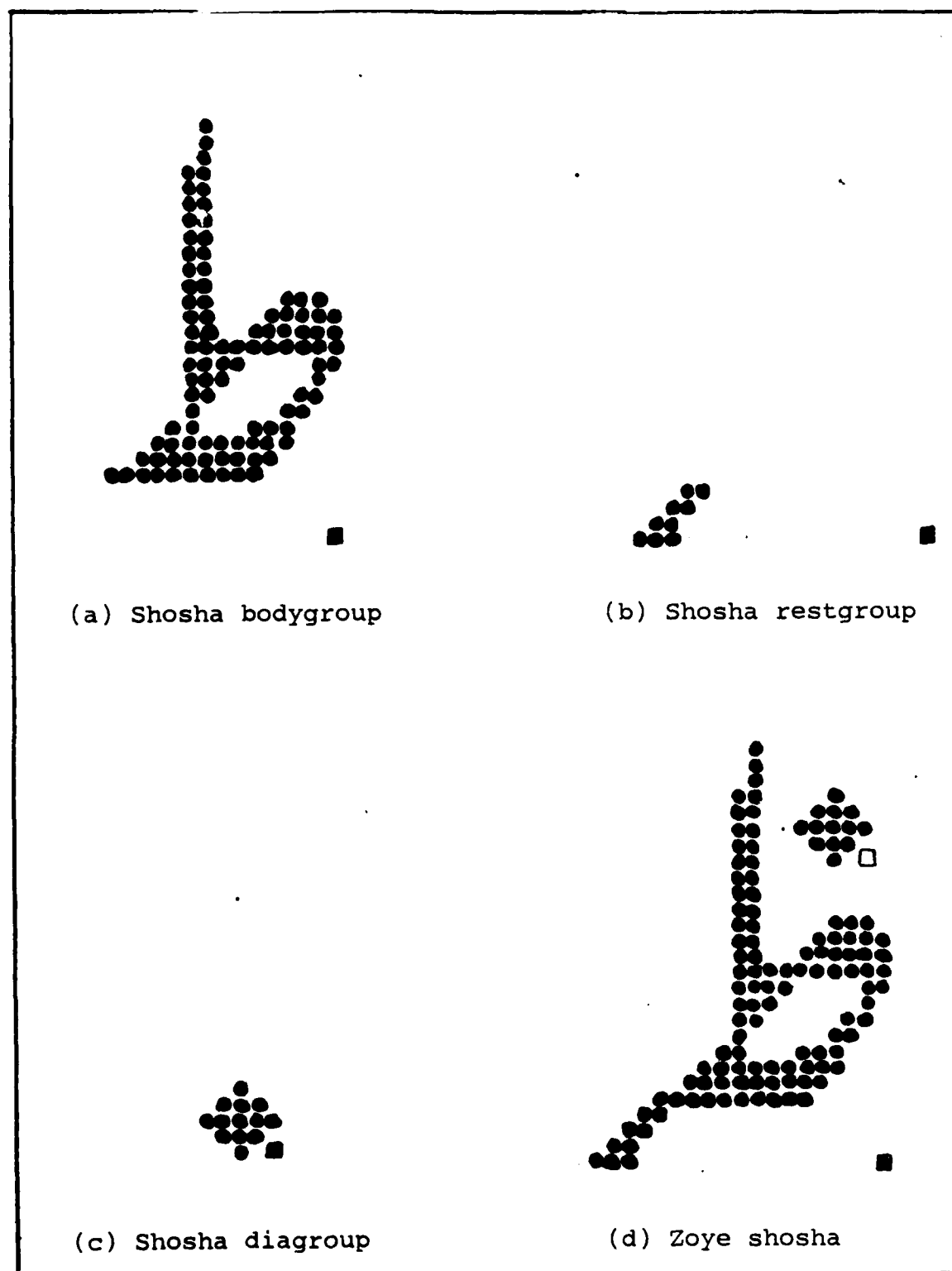


Figure 29. Elements of a Shosha Form of Zoye

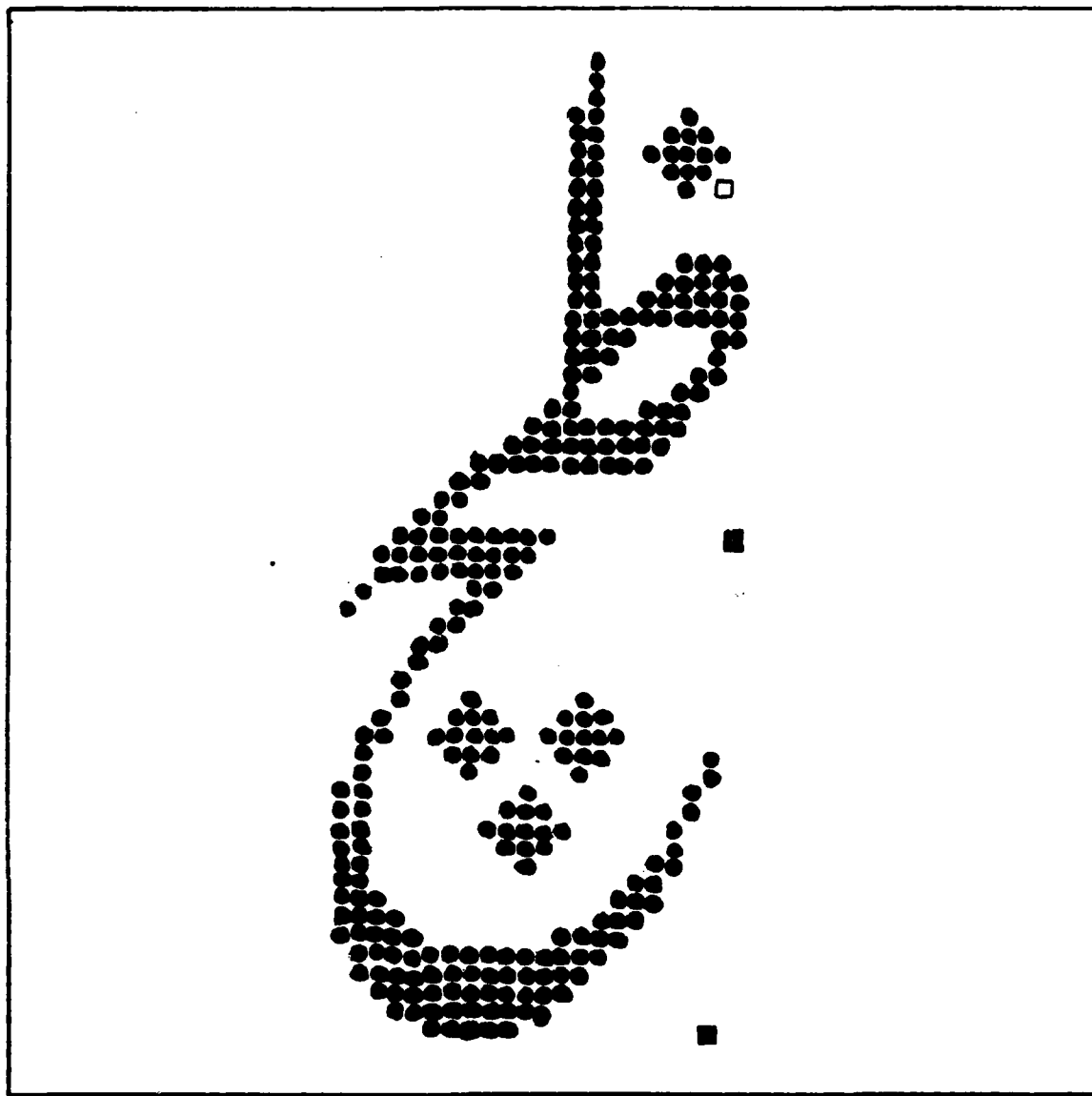


Figure 30. Zoye-Chey Ligature

Vita

Ronald Carl Durbin was born on 3 October 1951 in Wood River, Illinois. He is a 1969 graduate of East Alton-Wood River Community High School, and attended the University of Missouri at Rolla from which he received the degree of Bachelor of Science in Electrical Engineering in 1975. From 1969 through 1972 he was employed as an engineering co-op student at Mc Donnell-Douglas Corp. in St. Louis, Missouri. He enlisted in the Air Force in May 1973 and was commissioned from the Officer Training School in April 1976. He spent three years as an avionic systems engineer at the Air Force Avionics Laboratory, then returned to the Officer Training School as a flight commander in March 1979. He entered AFIT in March 1982. He is a Distinguished Graduate of both the Officer Training School and Squadron Officer School. Capt Durbin is married and has two children.

Permanent Address: 412 S. 9th Street
Wood River, IL 62095

Vita

Akeel Husain Kizilbash was born on 15th August, 1954 in Karachi, Pakistan. He earned his Secondary School Certificate from Islamabad Model School, Islamabad and completed his F.Sc. at the Islamic Science College, Karachi. He received the degree of Bachelor of Engineering in Avionics from the Pakistan Air Force College of Aeronautical Engineering, Karachi. Upon graduation in 1975, he joined the Armament Branch of the PAF. Since then he has worked in various technical and research positions as a maintenance officer, weapons specialist, and technical instructor. Before entering the School of Engineering, Air Force Institute of Technology, in 1982, he commanded the No. 109 Maintenance Unit of the PAF.

Permanent Address: 2/51-G P.E.C.H.S.
Karachi-29
Pakistan

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS			
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE						
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GEO/EE/83D-2			5. MONITORING ORGANIZATION REPORT NUMBER(S) AFIT/GEO/EE/83D-2			
6a. NAME OF PERFORMING ORGANIZATION Air Force Institute of Technology (AFIT)		6b. OFFICE SYMBOL (If applicable) EN	7a. NAME OF MONITORING ORGANIZATION Air Force Institute of Technology (AFIT)			
6c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433			7b. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Institute of Technology (AFIT)		8b. OFFICE SYMBOL (If applicable) EN	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER			
8c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433			10. SOURCE OF FUNDING NOS.			
11. TITLE (Include Security Classification) A SCHEME FOR THE COMPUTERIZED COMPOSITION			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
12. PERSONAL AUTHOR(S) Ronald C. Durbin, Capt, USAF and Akeel H. Kizilbash, Sqn. Ldr., PAF						
13a. TYPE OF REPORT MS Thesis		13b. TIME COVERED FROM 8303 TO 8312		14. DATE OF REPORT (Yr., Mo., Day) 831206		
				15. PAGE COUNT 119		
16. SUPPLEMENTARY NOTATION Approved for public release: LAW AFR 180-19. LYNN E. WOLFE 2 Feb 84 Dean for Research and Professional Development Air Force Institute of Technology (AFIT) Wright-Patterson AFB, OH 45433						
17. COSA CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)			
FIELD	GROUP	SUB GR.	Urdu Language Keyboard Design Nastaliq Computerized Text Composition Font Standardization			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) A scheme for the computerized composition of Urdu Nastaliq is described. A standardized character set was defined for the Urdu language. The script was decomposed into its constituent elements which were then defined and digitized in accordance with pattern recognition principles. An efficient mapping of Urdu characters to the standard QWERTY keyboard was achieved by matching character frequency to finger agility and by applying additional pattern recognition concepts. An analysis of finger work loads was then used to make final adjustments to the layout. A set of rules for combining characters was developed, by which a sizeable portion of the Nastaliq script could be composed. Rules for certain exceptional cases were not implemented. A program for displaying well-formed Nastaliq using a Vax 11/780 and a VT-550 graphics terminal was devised; the scheme is described in hardware-independent terms. Samples of text thus obtained illustrate the true composition, style, and elegance of the Nastaliq script. Recommendations for expanding the existing software to handle the entire language are included.						
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED			
22a. NAME OF RESPONSIBLE INDIVIDUAL			22b. TELEPHONE NUMBER (Include Area Code)		22c. OFFICE SYMBOL	

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

11. COMPOSITION OF URDU NASTALIQ (UNCLASSIFIED)

